McDATA®

Networking the world's business data™

SNMP

E/OSi SNMP Support Manual

P/N 620-000228-040
REV A

Simplifying Storage Network Management

McDATA Corporation
380 Interlocken Crescent Broomfield, CO 80021-3464
Corporate Headquarters: 800-545-5773
Sales E-mail: sales@mcdata.com *Web:* www.mcdata.com



Record of Revisions and Updates

Revision	Date	Description
620-000228-010	12/2004	Initial release of Manual
620-000228-020	01/2005	Updated version for E/OSi version 4.6.1
620-000228-030	06/2005	Updated version for E/OSi version 4.5
620-000228-040	10/2005	Updated version for E/OSi version 4.7

Printed October 2005
Fourth Edition

©2005 McDATA Corporation. All rights reserved.
McDATA, the McDATA logo, McDATA Eclipse, Fabricenter, HotCAT, Intrepid, Multi-Capable Storage Network Solutions, Networking the World's Business Data, nScale, nView, OPENready, SANavigator, SANpilot, SANtegrity, SANvergence, SecureConnect and Sphereon are trademarks or registered trademarks of McDATA Corporation. OEM and Reseller logos are the property of such parties and are reprinted with limited use permission. All other trademarks are the property of their respective companies. All specifications subject to change.
No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of McDATA Corporation.

The information contained in this document is subject to change without notice. McDATA Corporation assumes no responsibility for any errors that may appear.

All computer software programs, including but not limited to microcode, described in this document are furnished under a license, and may be used or copied only in accordance with the terms of such license. McDATA either owns or has the right to license the computer software programs described in this document. McDATA Corporation retains all rights, title and interest in the computer software programs.

McDATA Corporation makes no warranties, expressed or implied, by operation of law or otherwise, relating to this document, the products or the computer software programs described herein. McDATA CORPORATION DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE. In no event shall McDATA Corporation be liable for (a) incidental, indirect, special, or consequential damages or (b) any damages whatsoever resulting from the loss of use, data or profits, arising out of this document, even if advised of the possibility of such damages.

Contents

Preface vii

Chapter 1 Introduction to SNMP

SNMP Management 1-1
- Standard MIBs 1-4
- Private Enterprise MIBs 1-5
- Traps and Their Purpose 1-6

Chapter 2 McDATA SNMP Support

Overview 2-1
SAN Router Supported MIBs 2-2
- Proprietary MIB versions 2-4
- Standard MIB Support 2-5

TRAPS 2-9
- Fibre Alliance Traps 2-10

MIB Definitions 2-12
- MIB-II 2-12

Fabric Element Management MIB 2-53
- Fabric Element Management MIB Tables 2-53
- Class 2 Accounting table 2-80

FCMGMT-MIB 2-89
- The Sensor table 2-109
- The port table 2-113
- Event group 2-126
- Zone Alias Table 2-142
- Port Statistics 2-147
- The Fibre Channel Simple Name Server table 2-166
- Platform Table 2-172

SNMP trap registration group 2-177
Related traps 2-181

Appendix A McDATA/Nishan Common MIB

Nishan Enterprise MIB for FC Platform A-1
Common Group A-7
Nishan Extension MIB, Version Group A-7
Nishan Extension MIB, Authentication Group A-11
Nishan Extension MIB, Authenticating device logins A-18
Nishan Extension MIB, Trap Group A-23
Nishan Extension MIB, Load TFTP Group A-32
Architecture Group A-37
Interface Information A-37

Appendix B McDATA SAN Routing Management MIB

NISHAN-FCMGMT.MIB B-1
R_Port Support Branch B-2
R_Port Zoning Information B-15
R_Port Security Tables B-29
R_Port Route Tables B-36
R_Port Subfabric Tables B-39

Appendix C McDATA Gateway MIB

Nishan Gateway MIB (iFCP Gateway) V1 ..... C-1
Local SAN configuration C-3
iFCP Remote Peer Configuration C-10
Remote Connection Table C-19
iFCP Backup Link Info C-28

Appendix D McDATA iSCSI Configuration MIB

McDATA iSCSI Configuration MIB D-1
iSCSI Device Configuration D-3

Appendix E McDATA Eclipse SAN Router Management MIB

NISHAN-MGT.MIB E-1
Port configuration E-9
Management Port configuration E-9
System configuration E-14
Flash Memory Operations E-20
File Transfer Information E-23

Flash Operation Commands E-24
Environmental information E-27
FC Switch Zone Configuration E-30
ZONE INFO TABLE E-31
FC Port Map Table E-34
FC Storage Name Server group E-37
Naming Service's Port Table E-39
AL Port Map Table E-48
Naming Service's Node Table E-49
FC Name Server Configuration E-55
iSCSI LUN Mapping and Masking E-61
fcswLunInfoTable E-62
fcswLunMappingTable E-68
fcswSnsClientTable E-70
Chassis information E-74
Expansion Card Table E-81
Port Table E-84
FC Port Table E-91
FC R_Port Table E-96
TCP Port Table E-106
TCP Port Compression Statistics Table E-125
TCP Port Storage Statistics Table E-128
TCP Port Sessions Table E-131
TCP Sessions Storage Statistics Table E-134
Link Aggregation E-138

Appendix F McDATA SAN Routing Management MIB

FCMGMT-MIB Definitions F-1
Connectivity unit group F-4
Sensor table F-20
Port Table F-24
Event Group F-37
Link Table F-42
Port Statistics F-49
FC Simple Name Server Table F-67
SNMP Trap Registration Group F-73
Related Traps F-76

Appendix G McDATA/Nishan FC Fabric Element MIB

McDATA/Nishan FC Fabric Element Management MIB G-1
Configuration group G-6
The Module Table G-7

FxPort Configuration Table....G-11
FxPort common service parameters....G-14
FxPort class service parameters....G-16
Other FxPort parameters....G-17
Operation group....G-18
FxPort Operation table....G-18
The FxPort Fabric Login table....G-25
Error group....G-31
Accounting Groups....G-37
The Class 1 Accounting table....G-37
The Class 2 Accounting table....G-37
The Class 3 Accounting Group....G-41
Capability Group....G-44

Appendix H McDATA Management Traps MIB

McDATA Management Traps MIB....H-1
Eclipse Series traps....H-3
iFCP Gateway Traps....H-8

Appendix I McDATA Products Object Identifier Tree

NISHAN-PRODUCTS.mib....I-1

Appendix J McDATA SAN Router SMI Definitions

NISHAN-SMI.MIB....J-1

Appendix K McDATA SNTP Configuration MIB

SNTP Configuration....K-2

Index....i-1

Preface

This publication is part of the documentation suite that supports the McDATA® Eclipse™ 1620 and 2640 SAN Routers, referred to as SAN Router in this document.

Who Should Use This Manual

Use this publication if you are planning to use SNMP to manage the SAN Router.

The publications listed in *Related Publications* provide considerable information about both concepts and McDATA products.

Organization of This Manual

This publication is organized as follows:

- Chapter 1, *Introduction to SNMP*, provides an introduction and overview of Simple Network Management (SNMP) and its operation.
- Chapter 2, *McDATA SNMP Support*, describes specific information available through SNMP, especially the Management Information Bases (MIBs) that are supported and the SNMP traps generated by the SAN Router.
- Appendix A, *McDATA/Nishan Common MIB* lists the McDATA/Nishan SAN Router Common MIBs.
- Appendix B, *McDATA SAN Routing Management MIB* contains management objects for McDATA/Nishan Storage over IP Devices.
- Appendix C, *McDATA Gateway MIB* describes the McDATA/Nishan SAN Router iFCP Gateway MIBs.

- Appendix D, *McDATA iSCSI Configuration MIB* lists the McDATA/Nishan iSCSI Configuration MIB.
- Appendix E, *McDATA Eclipse SAN Router Management MIB* lists the McDATA/Nishan SAN Router Management MIB.
- Appendix F, *McDATA SAN Routing Management MIB* lists the FA FC Management MIB with text modified by Nishan Systems for compilation and trap integration purposes.
- Appendix G, *McDATA/Nishan FC Fabric Element MIB* lists the Fabric Element MIB modified by McDATA/Nishan for its SAN Router products.
- Appendix H, *McDATA Management Traps MIB* contains the McDATA/Nishan SAN Router Enterprise Traps.
- Appendix I, *McDATA Products Object Identifier Tree* describes the McDATA/Nishan SAN Router family and related Object Identifiers.
- Appendix J, *McDATA SAN Router SMI Definitions* describes McDATA/Nishan SAN Router SMI definitions and textual conventions.
- Appendix K, *McDATA SNTP Configuration MIB* describes McDATA/Nishan management objects for Nishan SNTP configuration.

Manual Updates

Check the McDATA web site at www.mcdata.com for possible updates or supplements to this manual.

Related Publications

Other publications that provide additional information about the products mentioned in this manual are:

- *SAN Routing E_Port and iFCP, Concepts and Technologies, Configuration Options, Design Guidelines, Best Practices, Caveats White Paper - by Prasad Pammidimukkala*
- *SANvergence Manager User Manual* (P/N 620-00189-150)
- *Eclipse 2640 SAN Router Administration and Configuration Manual* (P/N 620-00203)
- *E/OSi Command Line Interface User Manual (P/N 620-00207)*

- *Eclipse 1620 SAN Router Administration and Configuration Manual* (P/N 620-000206)
- *McDATA Products in a SAN Environment Planning Manual* (P/N 620-000124)
- *IP SANs, Tom Clark, Addison-Wesley,* ISBN 0-201-75277-8
- *Designing Storage Area Networks Second Edition, Tom Clark, Addison-Wesley,* ISBN 0-321-13650-0
- *Gigabit Ethernet: Technology and Applications for High-Speed LANs, Addison-Wesley,* ISBN 0-201-18553-9
- *Fibre Channel: A Comprehensive Introduction, NLA,* ISBN 0-931836-84-0
- *Basics of SCSI, Fourth Edition, Ancot Corporation,* ISBN 0-963-74398-8

Manual Conventions

The following notational conventions are used in this document.

Convention	Meaning
Italic	Outside book references, names of user interface windows, panels, buttons, and dialog boxes
Bold	Keyboard keys
Click. As in "click the icon on the navigation control panel."	Click with the left mouse button on the object to activate a function.
Right-click. As in "right click the product icon."	Click with the right mouse button on the object to activate a function.
Select. As in "select the log entry."	Click once on the object to highlight it.

Where to Get Help

For technical support, McDATA® end-user customers should call the phone number located on the service label attached to the front or rear of the hardware product.

McDATA's "Best in Class" Solution Center provides a single point of contact for customers seeking help. The Solution Center will research, explore, and resolve inquires or service requests regarding McDATA products and services. The Solution Center is staffed 24 hours a day, 7 days a week, including holidays.

NOTE: To expedite warranty entitlement, please have your product serial number available.

McDATA Corporation

380 Interlocken Crescent

Broomfield, CO 80021

Phone: **(800) 752-4572** or **(720) 558-3910**

Fax: (720) 558-3851

E-mail: support@mcdata.com

NOTE: Customers who purchased the hardware product from a company other than McDATA should contact that company's service representative for technical support.

Forwarding Publication Comments

We sincerely appreciate any comments about this publication. Did you find this manual easy or difficult to use? Did it lack necessary information? Were there any errors? Could its organization be improved?

Please send your comments via e-mail, our home page, or FAX. Identify the manual, and provide page numbers and details. Thank you.

E-mail:	pubsmgr@mcdata.com
Home Page:	http://www.mcdata.com
FAX:	Technical Communications Manager (720) 558-8999

Ordering Publications

To order a paper copy of this manual, submit a purchase order as described in *Ordering McDATA Documentation Instructions*, which is found on McDATA's web site, http://www.mcdata.com. To obtain documentation CD-ROMs, contact your sales representative.

Trademarks

The following terms, indicated by a registered trademark symbol (®) or trademark symbol (™) on first use in this publication, are trademarks of McDATA Corporation in the United States, other countries, or both:

Registered Trademarks	Trademarks
Fabricenter®	E/OS™
HotCAT®	Eclipse™
Intrepid®	Fibre Channel Director™
McDATA®	OPENconnectors™
OPENready®	SANvergence™
SANavigator®	Sphereon™
SANpilot®	
SANtegrity®	

All other trademarked terms, indicated by a registered trademark symbol (®) or trademark symbol (™) on first use in this publication, are trademarks of their respective owners in the United States, other countries, or both.

1

Introduction to SNMP

Network management is a broad term, including workstation configuration, assignment of IP addresses, network design, architecture, network security, and topologies. All this can fall within the scope of a network manager.

Any protocol for managing networks must allow virtually all network devices and systems to communicate statistics and status information to network management stations (network managers). This communication must be independent of the primary network transmission medium and impose little effect on the efficiency of the network. Network managers must be able to obtain status information from managed devices, and make changes in the way the managed devices handle network traffic. Simple Network Management Protocol (SNMP) is one way of meeting these requirements.

SNMP Management

SNMP is a mechanism for network management that is complete, but simple. It is designed on the manager/agent paradigm, with the agent residing in the managed device. Information is exchanged between agents (devices on the network being managed) and managers (devices on the network through which management is done).

There are many possible transactions between agents and managers. These transactions vary widely with the different types of devices that can be managed. With so many varied requirements for reporting and management, the list of commands a manager must be

able to issue is overwhelming, and every new manageable device can increase the list. SNMP was created to allow all these things to be easily done on any growing network.

SNMP operates on a simple fetch/store concept. With SNMP the available transactions between manager and agent are limited to a handful. The manager can request information from the agent or modify variables on the agent. The agent can respond to a request by sending information, or if enabled to do so, voluntarily notify the manager of a change of status on the agent (issue a trap).

With SNMP, administrators can manage the Director configuration, faults, performance, accounting, and security from remote SNMP management stations.

SNMP Simplified

SNMP is the only network management protocol that is widely available from many vendors of TCP/IP networks and internetworks.

SNMP:

- Allows network management with a simple set of commands.
- Allows new devices added to a network to be easily managed with minimal intervention.
- Is adequate for many basic network management needs.
- Is generalized for application to networks other than TCP/IP, such as IPX and OSI.
- Provides considerable versatility for managing a great many types of devices.
- Allows all networks to employ the same method for management.

SNMP Commands

A manager requests information from an agent by sending a single command, the Get command. The Get command also has a variation (GetNextRequest) that permits more efficient operation:

- GetRequest – Requests the value of a specified variable on the agent. This command is used to retrieve management data.
- GetNextRequest – Requests the value of the next variable after the one specified in the command. This command is used to retrieve lists and tables of management data.

An agent responds to a request by sending a single command, the GetResponse command. This command contains the requested information.

A manager changes information (variables) in the agent by sending a single command, the SetRequest command. This command is used to manipulate management data.

A trap is used by an agent to report extraordinary events. Refer to *Traps and Their Purpose* on page 1-6.

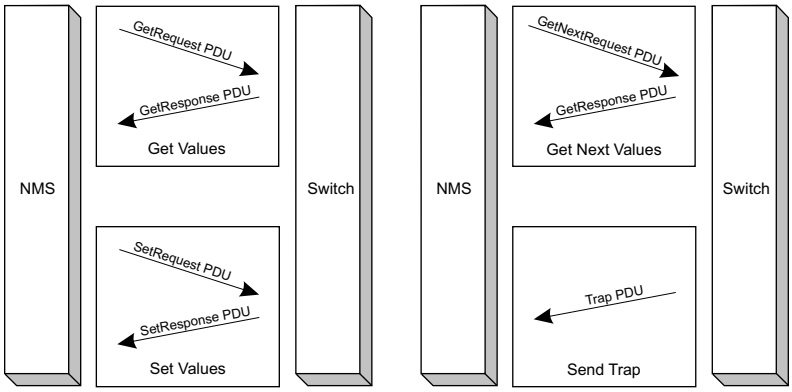


Figure 1-1 SNMP Commands and Responses

Why Variables Exist In a Managed Device

Variables are the means by which the Director (and other devices) keep track of their performance, control their own performance, and provide access to their performance for network managers. A simple example of a variable's use is to set a port offline and turn the port back on. Some variables just hold values that indicate status (for example error counts). SNMP allows the network managers to have access to some of the same variables for network management.

For purposes of the following explanation, an object is a data variable that represents an attribute of a managed device.

How SNMP Changes Variables (Objects) in a Managed Device

An agent is the entity that interfaces to the actual object being managed (Figure 1-2 on page 1-4). The agent understands the language of SNMP and translates between the manager and the object. Objects may be retrieved and/or modified by the manager, and it is the agent's job to return the requested object's value. Within the agent is at least one, maybe several, collections of definitions

called Management Information Bases (MIBs). When an agent supports a standard MIB, it agrees to provide and make available the variables listed in the MIB.

A MIB is a hierarchical tree of groups and variables. Operators at a network management station enter a command with supported groups and variables from the MIB.

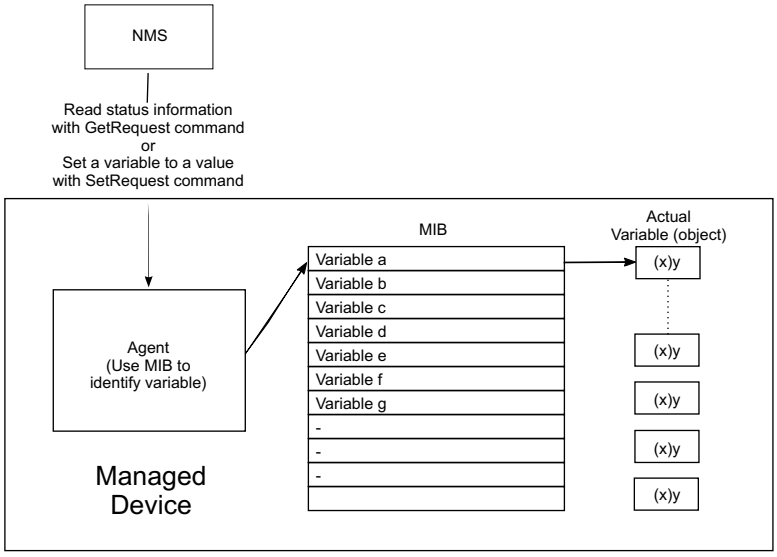


Figure 1-2 Retrieving or Setting Values Using MIBs

Standard MIBs

Standard MIBs are those created and approved by IETF and other Internet standards bodies and are readily available for use with SNMP network management stations. The standard MIBs provide a baseline of common operations across a wide variety of managed devices. Standard MIBs supported by the McDATA E/OSi are:

- MIB-II (Internet MIB) as described in RFC 1213
- IP Forwarding Table MIB as described in RFC 1354
- Bridge MIB as described in RFC 1493 that defines Managed Objects for Bridges
- Remote Network Monitoring MIB, as described in RFC 1757 that defines objects for managing remote network monitoring devices

- Fibre Alliance Fibre Channel Management MIB (FCMGMT) MIB, version 3.0
- Fibre Channel Fabric Element (FCFE), version 1.10
- P-Bridge MIB, the Bridge MIB Extension module for managing Priority and Multicast Filtering
- Q-Bridge MIB, the VLAN Bridge MIB module for managing Virtual Bridged Local Area Networks.

Private Enterprise MIBs

Private MIBs are those provided by the manufacturer of the managed devices to allow management of device-specific items. Chapter 2 describes the McDATA private Enterprise MIBs in more detail.

The McDATA private enterprise MIBs used by the McDATA E/OSi are:

- McDATA/Nishan Common MIB that lists the McDATA/Nishan SAN Router Common MIBs.
- McDATA SAN Routing Management MIB that contains management objects for McDATA/Nishan Storage over IP Devices.
- McDATA Gateway MIB that describes the McDATA/Nishan SAN Router iFCP Gateway MIBs.
- McDATA iSCSI Configuration MIB that lists the McDATA/Nishan iSCSI Configuration MIB.
- McDATA Eclipse SAN Router Management MIB that lists the McDATA/Nishan SAN Router Management MIB.
- McDATA/Nishan FC Fabric Element MIB that lists the Fabric Element MIB modified by McDATA/Nishan for its SAN Router products.
- McDATA Management Traps MIB that contains the McDATA/Nishan SAN Router Enterprise Traps.
- McDATA Products Object Identifier Tree that describes the McDATA/Nishan SAN Router family and related Object Identifiers.

- McDATA SAN Router SMI Definitions that describe McDATA/Nishan SAN Router SMI definitions and textual conventions.
- McDATA SNTP Configuration MIB describes management objects for Nishan SNTP configuration.

Traps and Their Purpose

Traps are unsolicited status reports, or status change indicators a managed object sends to a network manager. The destination address for traps is a configuration item for each managed agent. If configured, the SAN Routers send all traps as they occur.

2

McDATA SNMP Support

Overview

SNMP is a protocol that uses the User Data Protocol (UDP) to exchange messages between an SNMP agent (in a managed device) and a management station residing on a network. Although SNMP can be made available over other protocols, McDATA only supports UDP.

To be monitored and managed remotely by a network management station, each SAN Router is equipped with an SNMP agent. This agent is a software process within the SAN Router that receives management requests and generates corresponding responses by accessing the data specified for the MIB-II, Fabric Element MIB, Fibre Alliance MIB, and Enterprise MIB. In addition, the agent gives each SAN Router the ability to notify a management station when an important event occurs by sending a trap to the management station.

SAN Router Supported MIBs

This section describes the enterprise and standard MIBs supported by the SAN Router. The following table indicates the MIB files supported by each SAN Router.

A "Y" in the product column indicates that the MIB is supported. A "P" in the product column indicates that parts of the MIB are supported.

The SAN Router supports only SNMP version 1, not versions 2 or 3. The default read-only community string is "public", while1 the default read-write string is "private". At the same time, users can change the read-write community string.

1620	3300	4300	2640	MIB	Description
Y	Y	Y	Y	NISHAN-SMI.mib (McDATA SAN Router SMI Definitions)	Defines the "nishan" enterprise MIB branch
Y	Y	Y	Y	NISHAN-PRODUCTS.mib (McDATA Products Object Identifier Tree)	Defines product IDs
Y	Y	Y	Y	NISHAN-MGT.mib (McDATA Eclipse SAN Router Management MIB)	Eclipse Enterprise-specific MIB variables.
Y	Y	Y	Y	NISHAN-GTWY.mib (McDATA Gateway MIB)	iFCP gateway support
Y	Y	Y	Y	NISHAN-COM.mib (McDATA/Nishan Common MIB)	Eclipse Enterprise MIB variables.
Y	Y	Y	Y	NISHAN-ISCSI.mib (McDATA iSCSI Configuration MIB)	Eclipse iSCSI configuration MIB
Y	Y	Y	Y	NISHAN-FC.mib (McDATA/Nishan FC Fabric Element MIB)	Fabric Element MIB, similar to RFC 2837.
Y	Y	Y	Y	NISHAN-FCMGMT.mib (McDATA SAN Routing Management MIB)	Eclipse enterprise R_Port MIB

1620	3300	4300	2640	MIB	Description
P	P	P	P	NISHAN-RPRO.mib -	Eclipse inband address and static routes.
Y	Y	Y	Y	NISHAN-SNTP.mib	Eclipse enterprise SNTP MIB.
Y	Y	Y	Y	rfc1213.mib	MIB-II
Y	Y	Y	Y	rfc1354.mib	IP Forwarding table MIB
Y	Y	Y	Y	rfc1493.mib	Bridge MIB
P	P	P	P	rfc1757.mib	RMON MIB (4 tables supported)
P	P	P	P	P-BRIDGE-MIB.mib	802.1d/p Traffic Prioritization MIB from rfc2674
N	P	P	P	Q-BRIDGE-MIB.mib	802.1q VLAN Bridge MIB from rfc2674
Y	Y	Y	Y	FAFCMGMT.mib	Fibre Alliance MIB v3.0
Y	Y	Y	Y	NISHAN-TRAPS-v1.mib (McDATA Management Traps MIB)	Nishan's enterprise-specific SNMP v1 traps

The proprietary MIBs reference textual conventions, types, or OIDs from some standard RFC MIBs. To compile the proprietary MIBs, the following standard MIBs may be required. Some MIB compilers will not require the SNMP MIBs.

rfc2578.mib	SNMPv2-SMI, for SNMP data types
rfc2579.mib	SNMPv2-TC, for SNMP textual conventions
rfc2851.mib	INET-ADDRESS-MIB, for IP address types
rfc2863.mib	IF-MIB, for textual conventions

Some MIB browsers require MIBs to be loaded in a certain order, so that dependencies are loaded first. For example, most McDATA proprietary MIBs reference NISHAN-SMI, so NISHAN-SMI should be loaded before other McDATA MIBs. The following list is one possible order that loads dependencies first.

rfc1213.mib	MIB-II
rfc1354.mib	IP Forwarding table MIB
rfc1493.mib	Bridge MIB
rfc1757.mib	RMON MIB (4 tables supported)
rfc2578.mib	SNMPv2-SMI, for SNMP data types
rfc2579.mib	SNMPv2-TC, for SNMP textual conventions
rfc2851.mib	INET-ADDRESS-MIB, for IP address types
rfc2863.mib	IF-MIB, for textual conventions
P-BRIDGE-MIB.mib	802.1d/p Traffic Prioritization MIB from rfc2674
FAFCMGMT.mib	Fibre Alliance MIB v3.0
NISHAN-SMI.mib	Defines the "nishan" enterprise MIB branch
NISHAN-PRODUCTS.mib	Defines product IDs
NISHAN-MGT.mib	Eclipse enterprise-specific MIB variables
NISHAN-GTWY.mib	iFCP gateway support
NISHAN-COM.mib	Eclipse common enterprise MIB variables
NISHAN-ISCSI.mib	Eclipse iSCSI configuration MIB
NISHAN-FC.mib	Fabric Element MIB, similar to RFC 2837.
NISHAN-FCMGMT.mib	Eclipse enterprise FC R_Port MIB
NISHAN-RPRO.mib	Eclipse inband address and static routes
NISHAN-SNTP.mib	Eclipse enterprise SNTP MIB
NISHAN-TRAPS-v1.mib	McDATA enterprise-specific SNMP v1 traps

Proprietary MIB versions

Proprietary MIB versions are identified by the LAST-UPDATED field in the MODULE-IDENTITY macro near the top of each MIB file. The version of each proprietary MIB supported in E/OSi release 4.7 is:

MIB	Last-Updated
NISHAN-SMI.mib	0509190000Z
NISHAN-PRODUCTS.mib	0509190000Z
NISHAN-MGT.mib	0509290000Z
NISHAN-GTWY.mib	0509190000Z

MIB	Last-Updated
NISHAN-COM.mib	0509190000Z
NISHAN-ISCSI.mib	0509190000Z
NISHAN-FC.mib	0509190000Z
NISHAN-FCMGMT.mib	0509190000Z
NISHAN-ISNS-MGT.mib	0509190000Z
NISHAN-SNTP.mib	0509190000Z
NISHAN-TRAPS-v1.mib	0509190000Z

Standard MIB Support

The degree of support for standard MIBs is defined in the tables below:

RFC1213 (MIB-II)

Ref	MIB Groups	Supported
1	system	Yes
2	interfaces	Yes
3	at	Yes
4	ip	Yes
5	icmp	Yes
6	tcp	Yes
7	udp	Yes
8	egp	No
9	transmission	No
10	snmp	Yes

RFC1354 (IP Forwarding)

Ref	MIB Groups	Supported
1	ipForward	Yes

RFC1493 (Bridge)

Ref	MIB Groups	Supported
1	dot1dBase	Yes
2	dot1dStp	Yes
3	dot1dSr	No
4.1	dot1dTpFdbTable	Yes
4.2	dot1dTpPortTable	No
6	dot1dStatic	No

RFC1757 (RMON)

Ref	MIB Groups	Supported
1	Statistics	Yes
2	History	Yes
3	Alarms	Yes
4	Host	No
5	Host Top	No
6	Matrix	No
7	Filter	No
8	Capture	No
9	Event	Yes

RFC2674 (802.1p)

Ref	MIB Groups	Supported
1	dot1dExtBase	Yes
2.1	dot1dPortPriorityTable	Yes
2.2	dot1dUserPriorityRegenTable	No
2.3	dot1dTrafficClassTable	Yes
2.4	dot1dPortOutboundAccessPriorityTable	No
3	dot1dGarp	Yes
4	dot1dGmrp	Yes
5.1	dot1dTpHCPortTable	No
5.2	dot1dTpPortOverflowTable	No

FAFCMGMT.mib - Fibre Alliance MIB v3.0

Ref	MIB Groups	Supported
1	uNumber	Yes
2	systemURL	Yes
3	statusChangeTime	No (obsolete)
4	configurationChangeTime	No (obsolete)
5	connUnitTableChangeTime	No (obsolete)
6	connUnitTable	Yes
7	connUnitRevsTable	Yes
8	connUnitSensorTable	Yes
10	connUnitPortTable	Yes
11	connUnitEventTable	No
12	connUnitLinkTable	Yes
2.3	trapRegTable	No

Ref	MIB Groups	Supported
3	revisionNumber	Yes
4.5	connUnitPortStatTable	Yes
5.2	connUnitSnsTable	No

TRAPS

Traps generated by the SAN Router are:

1620	2640	
P	P	Generic SNMP traps
N	Y	Spanning Tree traps
Y	Y	RMON traps
P	Y	Nishan enterprise-specific traps

Generic Traps (from RFC1157)

Ref	MIB	Supported
0	coldStart	switch startup
2	linkDown	port down
3	linkUp	port up
4	authenticationFailure	invalid community string

RMON traps

From RFC1757, enterprise = 1.3.6.1.2.1.16

Ref	MIB	Supported
1	risingAlarm	A monitored value exceeded its rising alarm threshold
2	fallingAlarm	A monitored value dropped below its falling alarm threshold

Fibre Alliance Traps

From FAFCMGMT MIB, enterprise = 1.3.6.1.3.94

Ref	Object
1	connUnitStatusChange
6	connUnitPortStatusChange

NOTE: For connUnitPortStatusChange, the implementation doesn't conform to IETF standards. To conform to the standard, it should report the port index in the varvbind list. Rather it reports the physical port, making it easier for the management application to correlate the port index to the physical port number.

Nishan Enterprise-specific Traps

(from NISHAN-TRAPS-v1 MIB, enterprise = 1.3.6.1.4.1.4369.3)

Ref	Object	Description
1	nSswSNSTrap	Switch has become primary mSNS server
3	nSswFlashOperationTrap	New software image written to flash
5	nSswEnvVoltageUpperThreshTrap	A power-supply voltage too high
6	nSswEnvVoltageLowerThreshTrap	A power-supply voltage too low
7	nSswEnvTempUpperThreshTrap	Internal temperature too high
8	nSswEnvChasFanStatusTrap	Fan failed
9	nSswEnvChasPowerSupplyStatusTrap	Power supply failed or restored
10	nSswLaAggDownTrap	Link aggregator no longer functioning (not in Model 1620)
11	nSswLaAggUpTrap	Link aggregator restored (not in Model 1620)
14	ifcpBackupDown	Port redundancy failed
15	ifcpBackupActivated	Backup link activated
16	ifcpRemoteConnectionUp	Inter-SAN link established

Ref	Object	Description
17	ifcpRemoteConnectionDown	Inter-SAN link failed
18	rPortConfigurationChanged	R_Port configuration changed
19	rPortZoningChanged	FC switch zoning changed by the SAN Router
20	rPortFailedtoEstablishISL	R_Port connection failed
21	rPortFabricReconfiguration	FC fabric topology change
22	rPortFabricCriticalError	R_Port critical fabric error

MIB Definitions

MIB-II

There are eleven groups of objects specified in MIB-II. The E/OSi SNMP Agent supports eight groups:

- *System Group* on page 2-12. This group provides general information about the managed system.
- *Interfaces Group* on page 2-15.
- *Address Translation Group* on page 2-22. This group is implemented, but the corresponding table may be empty.
- *IP Group* on page 2-23.
- *ICMP Group* on page 2-35.
- *TCP Group* on page 2-39
- *UDP Group* on page 2-45.
- *SNMP Group* on page 2-46. This group keeps statistics on the SNMP Agent implementation itself.

System Group

sysDescr

Type	DisplayString(0..255)
Access	R
Description	A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating system, and networking software. It is mandatory that this only contain printable ASCII characters.

sysObjectID

Type	Object Identifier
Access	R
Description	The vendor’s authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining ‘what kind of box’ is being

managed. For example, if vendor 'Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its 'Fred Router'.

The current set of sysObjectID values recognized by Element Manager v4.6 with the corresponding vendor and model names are given in the table below:

sysObjectID	Vendor name	Model Name	Supported?
1.3.6.1.4.1.4369.3.2.1.1	McDATA Corporation	Eclipse 3100	No
1.3.6.1.4.1.4369.3.2.1.2	McDATA Corporation	Eclipse 3300	Yes
1.3.6.1.4.1.4369.3.2.1.3	McDATA Corporation	Eclipse 3320	No
1.3.6.1.4.1.4369.3.2.1.4	McDATA Corporation	Eclipse 1620	Yes
1.3.6.1.4.1.4369.3.2.2.1	McDATA Corporation	Eclipse 4100	No
1.3.6.1.4.1.4369.3.2.2.2	McDATA Corporation	Eclipse 4300	Yes
1.3.6.1.4.1.4369.3.2.3.1	McDATA Corporation	Eclipse 2600	No
1.3.6.1.4.1.4369.3.2.3.2	McDATA Corporation	Eclipse 2640	Yes
1.3.6.1.4.1.4369.3.2.3.3	McDATA Corporation	Eclipse 2680	No

sysUpTime

Type TimeTicks

Access R

Description The time (in hundredths of a second) since the network management portion of the system was last re-initialized.

sysContact

Type DisplayString (0..255)

Access R

Description The textual identification of the contact person for this managed node, together with information on how to contact this person.

sysName

Type	DisplayString (0..255)
Access	RW
Description	An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name.

sysLocation

Type	DisplayString (0..255)
Access	RW
Description	The physical location of this node (e.g., `telephone closet, 3rd floor').

sysServices

Type	INTEGER
Access	R
Description	A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero, then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a node which performs primarily routing functions would have a value of 4 (2^(3-1)). In contrast, a node which is a host offering application services would have a value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly: layer functionality 1 physical (e.g., repeaters) 2 datalink/subnetwork (e.g., bridges) 3 internet (e.g., IP gateways) 4 end-to-end (e.g., IP hosts) 7 applications (e.g., mail relays) For systems including OSI protocols, layers 5 and 6 may also be counted.

Interfaces Group

ifNumber

Type	INTEGER
Access	R
Description	The number of network interfaces (regardless of their current state) present on this system.

The Interfaces Table

The interfaces table contains information on the entity's interfaces. Each interface is thought of as being attached to a “subnetwork”. Note that this term should not be confused with “subnet” which refers to an addressing partitioning scheme used in the Internet suite of protocols.

---

ifIndex

Type INTEGER

Access R

Description A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

---

ifDescr

Type DisplayString(0..255)

Access R

Description A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.

---

ifType

Type INTEGER

Access R

Description The type of interface, distinguished according to the physical/link protocol(s) immediately below the network layer in the protocol stack.

Values:

other(1), none of the following

regular1822(2),

hdh1822(3),

ddn-x25(4),

rfc877-x25(5),

ethernet-csmacd(6),
iso88023-csmacd(7),
iso88024-tokenBus(8),
iso88025-tokenRing(9),
iso88026-man(10),
starLan(11),
proteon-10Mbit(12),
proteon-80Mbit(13),
hyperchannel(14),
fddi(15),
lapb(16),
sdlc(17),
ds1(18), T-1
e1(19), european equivalent of T-1
basicISDN(20),
primaryISDN(21), proprietary serial
propPointToPointSerial(22),
ppp(23),
softwareLoopback(24),
eon(25) CLNP over IP [11]
ethernet-3Mbit(26),
nsip(27), XNS over IP
slip(28), generic SLIP
ultra(29), ULTRA technologies
ds3(30), T-3
sip(31), SMDS
frame-relay(32)

ifMtu

Type	INTEGER
Access	R
Description	The size of the largest datagram which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

ifSpeed

Type	Gauge
Access	R
Description	An estimate of the interface's current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.

ifPhysAddress

Type	PhysAddress
Access	R
Description	The interface's address at the protocol layer immediately `below' the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

ifAdminStatus

Type	INTEGER
Access	RW
Description	The desired state of the interface. The testing(3) state indicates that no operational packets can be passed.

ifOperStatus

Type	INTEGER
Access	R

Description The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed.

ifLastChange

Type TimeTicks

Access R

Description The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a zero value.

ifInOctets

Type Counter

Access R

Description The total number of octets received on the interface, including framing characters.

ifInUcastPkts

Type Counter

Access R

Description The number of subnetwork-unicast packets delivered to a higher-layer protocol.

ifInNUcastPkts

Type Counter

Access R

Description The number of non-unicast (i.e., subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.

ifInDiscards

Type Counter

Access R

Description The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

ifInErrors

Type Counter

Access R

Description The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

ifInUnknownProtos

Type Counter

Access R

Description The number of packets received via the interface which were discarded because of an unknown or unsupported protocol

ifOutOctets

Type Counter

Access R

Description The total number of octets transmitted out of the interface, including framing characters.

ifOutUcastPkts

Type Counter

Access R

Description The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.

ifOutNUcastPkts

Type Counter

Access R

Description The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.

ifOutDiscards

Type Counter

Access R

Description The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

ifOutErrors

Type Counter

Access R

Description The number of outbound packets that could not be transmitted because of errors.

ifOutQLen

Type Gauge

Access R

Description The length of the output packet queue (in packets).

ifSpecific

Type OBJECT IDENTIFIER

Access R

Description A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an ethernet, then the value of this object refers to a document defining objects specific to ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier, and any conforming

implementation of ASN.1 and BER must be able to generate and recognize this value.

Address Translation Group

Implementation of the Address Translation group is mandatory for all systems. Note however that this group is deprecated by MIB-II. That is, it is being included solely for compatibility with MIB-I nodes, and will most likely be excluded from MIB-III nodes. From MIB-II and onwards, each network protocol group contains its own address translation tables.

The Address Translation group contains one table which is the union across all interfaces of the translation tables for converting a NetworkAddress (e.g., an IP address) into a subnetwork-specific address. For lack of a better term, this document refers to such a subnetwork-specific address as a `physical' address. Examples of such translation tables are: for broadcast media where ARP is in use, the translation table is equivalent to the ARP cache; or, on an X.25 network where non-algorithmic translation to X.121 addresses is required, the translation table contains the NetworkAddress to X.121 address equivalences.

atIfIndex

Type	INTEGER
Access	RW
Description	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

atPhysAddress

Type	PhysAddress
Access	RW
Description	The media-dependent `physical' address. Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively disassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not

currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

atNetAddress

Type NetworkAddress

Access RW

Description The NetworkAddress (e.g., the IP address) corresponding to the media-dependent `physical' address.

IP Group

ipForwarding

Type INTEGER

Access RW

Description The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not (except those source-routed via the host). Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a `badValue' response if a management station attempts to change this object to an inappropriate value.

ipDefaultTTL

Type INTEGER

Access RW

Description The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol.

ipInReceives

Type Counter

Access R

Description The total number of input datagrams received from interfaces, including those received in error.

ipInHdrErrors

Type Counter

Access R

Description The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, Time-To-Live exceeded, errors discovered in processing their IP options, etc.

ipInAddrErrors

Type Counter

Access R

Description The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported classes (e.g., Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address.

ipForwDatagrams

Type Counter

Access R

Description The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source-Route option processing was successful.

ipInUnknownProtos

Type Counter

Access R

Description The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol.

ipInDiscards

Type Counter

Access R

Description The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded (e.g., for lack of buffer space). Note that this counter does not include any datagrams discarded while awaiting re-assembly.

ipInDelivers

Type Counter

Access R

Description The total number of input datagrams successfully delivered to IP user-protocols (including ICMP).

ipOutRequests

Type Counter

Access R

Description The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in `ipForwDatagrams`.

ipOutDiscards

Type Counter

Access R

Description The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (e.g., for lack of buffer space). Note that this counter would include datagrams counted in `ipForwDatagrams` if any such packets met this (discretionary) discard criterion.

ipOutNoRoutes

Type	Counter
Access	R
Description	The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in `ipForwDatagrams` which meet this `no-route' criterion. Note that this includes any datagrams which a host cannot route because all of its default gateways are down.

ipReasmTimeout

Type	INTEGER
Access	R
Description	The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity.

ipReasmReqds

Type	Counter
Access	R
Description	The number of IP fragments received which needed to be reassembled at this entity.

ipReasmOKs

Type	Counter
Access	R
Description	The number of IP datagrams successfully.

ipReasmFails

Type	Counter
Access	R
Description	The number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, etc). Note that this is not necessarily a count of discarded IP fragments since some algorithms

(notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received.

ipFragOKs

Type Counter

Access R

Description The number of IP datagrams that have been successfully fragmented at this entity.

ipFragFails

Type Counter

Access R

Description The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, e.g., because their Don't Fragment flag was set.

ipFragCreates

Type Counter

Access R

Description The number of IP datagram fragments that have been generated as a result of fragmentation at this entity.

The IP Address Table The IP address table contains this entity's IP addressing information.

ipAdEntAddr

Type IpAddress

Access R

Description The IP address to which this entry's addressing information pertains.

ipAdEntIfIndex

Type INTEGER

Access R

Description The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

ipAdEntNetMask

Type IpAddress

Access R

Description The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.

ipAdEntBcastAddr

Type INTEGER

Access R

Description The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface.

ipAdEntReasmMaxSize

Type INTEGER (0..65535)

Access R

Description The size of the largest IP datagram which this entity can re-assemble from incoming IP fragmented datagrams received on this interface.

The IP routing group

The IP routing group contains an entry for each route presently known to this entity.

ipRouteDest

Type IpAddress

Access RW

Description The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.

ipRouteIfIndex

Type INTEGER

Access RW

Description The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

ipRouteMetric1

Type INTEGER

Access RW

Description The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteMetric2

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteMetric3

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteMetric4

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteNextHop

Type	IpAddress
Access	RW
Description	The IP address of the next hop of this route. (In the case of a route bound to an interface which is realized via a broadcast media, the value of this field is the agent's IP address on that interface.)

ipRouteType

Type INTEGER

Access RW

Description The type of route. Note that the values direct(3) and indirect(4) refer to the notion of direct and indirect routing in the IP architecture.

Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively disassociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

Values:

other(1),	none of the following
invalid(2),	an invalidated route
direct(3),	route to directly connected (sub-)network
indirect(4)	route to a non-localhost/network/sub-network

ipRouteProto

Type INTEGER

Access R

Description The routing mechanism via which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.

other(1)	none of the following
local(2)	-protocol information, e.g., manually configured entries
netmgmt(3)	set via a network management protocol
icmp(4)	e.g., obtained via ICMP, Redirect

The remaining values are all gateway routing protocols:

egp(5),

ggp(6),

hello(7),

rip(8),

is-is(9),

es-is(10),

ciscoIgrp(11),

bbnSpfIgp(12),

ospf(13),

bgp(14)

ipRouteAge

Type INTEGER

Access RW

Description The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of `too old' can be implied except through knowledge of the routing protocol by which the route was learned.

ipRouteMask

Type IpAddress

Access RW

Description Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belong to a class-A, B, or C network, and then using one of:

mask	network
255.0.0.0	class-A
255.255.0.0	class-B
255.255.255.0	class-C

If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism.

ipRouteMetric5

Type INTEGER

Access RW

Description An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteInfo

Type OBJECT IDENTIFIER

Access R

Description A reference to MIB definitions specific to the particular routing protocol which is responsible for this route, as determined by the value specified in the route's ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0}, which is a syntactically valid object identifier, and any conforming implementation of ASN.1 and BER must be able to generate and recognize this value.

IP Address Translation Table

The IP address translation table contain the IpAddress to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (e.g., DDN-X.25 has an algorithmic method); if all interfaces are of this type, then the Address Translation table is empty, i.e., has zero entries.

ipNetToMediaIfIndex

Type INTEGER

Access RW

Description The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

ipNetToMediaPhysAddress

Type	PhysAddress
Access	RW
Description	The media-dependent `physical' address.

ipNetToMediaNetAddress

Type	IpAddress
Access	RW
Description	The IpAddress corresponding to the media-dependent `physical' address

ipNetToMediaType

Type	INTEGER
Access	RW
Description	The type of mapping. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively disassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant `ipNetToMediaType` object.

Values:

other(1)	none of the following
invalid(2)	an invalidated mapping
dynamic(3)	
static(4)	

Additional IP objects

ipRoutingDiscards

Type	Counter
Access	R
Description	The number of routing entries which were chosen to be discarded even though they are valid. One possible reason for discarding such an entry could be to free up buffer space for other routing entries.

ICMP Group

icmpInMsgs

Type	Counter
Access	R
Description	The total number of ICMP messages which the entity received. Note that this counter includes all those counted by icmpInErrors.

icmpInErrors

Type	Counter
Access	R
Description	The number of ICMP messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.).

icmpInDestUnreachs

Type	Counter
Access	R
Description	The number of ICMP Destination Unreachable messages received.

icmpInTimeExcds

Type	Counter
Access	R
Description	The number of ICMP Time Exceeded messages received.

icmpInParmProbs

Type	Counter
Access	R
Description	The number of ICMP Parameter Problem messages received.

icmpInSrcQuenchs

Type	Counter
Access	R
Description	The number of ICMP Source Quench messages received.

icmpInRedirects

Type	Counter
Access	R
Description	The number of ICMP Redirect messages received.

icmpInEchos

Type	Counter
Access	R
Description	The number of ICMP Echo (request) messages received.

icmpInEchoReps

Type	Counter
Access	R
Description	The number of ICMP Echo Reply messages received.

icmpInTimestamps

Type	Counter
Access	R
Description	The number of ICMP Timestamp (request) messages received.

icmpInTimestampReps

Type	Counter
Access	R
Description	The number of ICMP Timestamp Reply messages received.

icmpInAddrMasks

Type	Counter
Access	R
Description	The number of ICMP Address Mask Request messages received.

icmpInAddrMaskReps

Type	Counter
Access	R
Description	The number of ICMP Address Mask Reply messages received

icmpOutMsgs

Type	Counter
Access	R
Description	The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors.

icmpOutErrors

Type	Counter
Access	R
Description	The number of ICMP messages which this entity did not send due to problems discovered within ICMP such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations there may be no types of error which contribute to this counter's value.

icmpOutDestUnreachs

Type	Counter
Access	R
Description	The number of ICMP Destination Unreachable messages sent.

icmpOutTimeExcds

Type	Counter
Access	R
Description	The number of ICMP Time Exceeded messages sent.

icmpOutParmProbs

Type	Counter
Access	R
Description	The number of ICMP Parameter Problem messages sent.

icmpOutSrcQuenchs

Type	Counter
Access	R
Description	The number of ICMP Source Quench messages sent.

icmpOutRedirects

Type	Counter
Access	R
Description	The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects.

icmpOutEchos

Type	Counter
Access	R
Description	The number of ICMP Echo (request) messages sent.

icmpOutEchoReps

Type Counter

Access R

Description The number of ICMP Echo Reply messages sent.

icmpOutTimestamps

Type Counter

Access R

Description The number of ICMP Timestamp (request) messages sent.

icmpOutTimestampReps

Type Counter

Access R

Description The number of ICMP Timestamp Reply messages sent.

icmpOutAddrMasks

Type Counter

Access R

Description The number of ICMP Address Mask Request messages sent.

icmpOutAddrMaskReps

Type Counter

Access R

Description The number of ICMP Address Mask Reply messages sent.

TCP Group

Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

tcpRtoAlgorithm

Type INTEGER

Access R

Description The algorithm used to determine the timeout value used for retransmitting unacknowledged octets.

Values:

other(1),	none of the following
constant(2)	a constant rto
rsre(3)	MIL-STD-1778, Appendix B
vanj(4)	Van Jacobson's algorithm [10]

tcpRtoMin

Type INTEGER

Access R

Description The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793.

tcpRtoMax

Type INTEGER

Access R

Description The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793.

tcpMaxConn

Type INTEGER

Access R

Description The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1.

tcpActiveOpens

Type Counter

Access R

Description The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.

tcpPassiveOpens

Type Counter

Access R

Description The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.

tcpAttemptFails

Type Counter

Access R

Description The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.

tcpEstabResets

Type Counter

Access R

Description The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

tcpCurrEstab

Type	Gauge
Access	R
Description	The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

tcpInSegs

Type	Counter
Access	R
Description	The total number of segments received, including those received in error. This count includes segments received on currently established connections.

tcpOutSegs

Type	Counter
Access	R
Description	The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.

tcpRetransSegs

Type	Counter
Access	R
Description	The total number of segments retransmitted. That is, the number of TCP segments transmitted containing one or more previously transmitted octets.

The TCP Connection Table

The TCP connection table contains information about this entity's existing TCP connections.

tcpConnState

Type	INTEGER
Access	RW

Description The state of this TCP connection. The only value which may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a `badValue' response if a management station attempts to set this object to any other value. If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node, resulting in immediate termination of the connection. As an implementation-specific option, an RST segment may be sent from the managed node to the other TCP endpoint (note however that RST segments are not sent reliably).

Values:

closed(1),

listen(2),

synSent(3),

synReceived(4),

established(5),

finWait1(6),

finWait2(7),

closeWait(8),

lastAck(9),

closing(10),

timeWait(11),

deleteTCB(12)

tcpConnLocalAddress

Type IpAddress

Access R

Description The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used.

tcpConnLocalPort

Type	INTEGER (0..65535
Access	R
Description	The local port number for this TCP connection.

tcpConnRemAddress

Type	IpAddress
Access	R
Description	The remote IP address for this TCP connection.

tcpConnRemPort

Type	INTEGER (0..65535)
Access	R
Description	The remote port number for this TCP connection.

Additional TCP Objects

tcpInErrs

Type	Counter
Access	R
Description	The total number of segments received in error (e.g., bad TCP checksums)

tcpOutRsts

Type	Counter
Access	R
Description	The number of TCP segments sent containing the RST flag.

UDP Group

udpInDatagrams

Type Counter

Access R

Description The total number of UDP datagrams delivered to UDP users.

udpNoPorts

Type Counter

Access R

Description The total number of received UDP datagrams for which there was no application at the destination port.

udpInErrors

Type Counter

Access R

Description The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.

udpOutDatagrams

Type Counter

Access R

Description The total number of UDP datagrams sent from this entity.

The UDP Listener Table

The UDP listener table contains information about this entity's UDP end-points on which a local application is currently accepting datagrams.

udpLocalAddress

Type IpAddress

Access R

Description The local IP address for this UDP listener. In the case of a UDP listener which is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used.

---

udpLocalPort

Type INTEGER (0..65535)

Access R

Description The local port number for this UDP listener.

SNMP Group Some of the objects defined below will be zero-valued in those SNMP implementations that are optimized to support only those functions specific to either a management agent or a management station. In particular, it should be observed that the objects below refer to an SNMP entity, and there may be several SNMP entities residing on a managed node (e.g., if the node is hosting acting as a management station).

---

snmpInPkts

Type Counter

Access R

Description The total number of messages delivered to the SNMP entity from the transport service.

---

snmpOutPkts

Type Counter

Access R

Description The total number of SNMP messages which were passed from the SNMP protocol entity to the transport service.

---

snmpInBadVersions

Type Counter

Access R

Description The total number of SNMP messages which were delivered to the SNMP protocol entity and were for an unsupported SNMP version.

snmpInBadCommunityNames

Type	Counter
Access	R
Description	The total number of SNMP messages delivered to the SNMP protocol entity which used a SNMP community name not known to said entity.

snmpInBadCommunityUses

Type	Counter
Access	R
Description	The total number of SNMP messages delivered to the SNMP protocol entity which represented an SNMP operation which was not allowed by the SNMP community named in the messages.

snmpInASNParseErrs

Type	Counter
Access	R
Description	The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP messages.

snmpInTooBigs

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `tooBig.'

snmpInNoSuchNames

Type	Counter
Access	R

Description The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `noSuchName.'

snmpInBadValues

Type Counter

Access R

Description The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `badValue.'

snmpInReadOnlys

Type Counter

Access R

Description The total number valid SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `readOnly.' It should be noted that it is a protocol error to generate an SNMP PDU which contains the value `readOnly' in the error-status field, as such this object is provided as a means of detecting incorrect implementations of the SNMP.

snmpInGenErrs

Type Counter

Access R

Description The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `genErr.'

snmpInTotalReqVars

Type Counter

Access R

Description The total number of MIB objects which have been retrieved successfully by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs.

snmpInTotalSetVars

Type	Counter
Access	R
Description	The total number of MIB objects which have been altered successfully by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs.

snmpInGetRequests

Type	Counter
Access	R
Description	The total number of SNMP Get-Request PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInGetNexts

Type	Counter
Access	R
Description	The total number of SNMP Get-Next PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInSetRequests

Type	Counter
Access	R
Description	The total number of SNMP Set-Request PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInGetResponses

Type	Counter
Access	R
Description	The total number of SNMP Get-Response PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInTraps

Type	Counter
Access	R
Description	The total number of SNMP Trap PDUs which have been accepted and processed by the SNMP protocol entity.

snmpOutTooBigs

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is `tooBig.'

snmpOutNoSuchNames

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status is `noSuchName.'

snmpOutBadValues

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is `badValue.'

snmpOutGenErrs

Type	Counter
Access	R

Description The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is `genErr.'

snmpOutGetRequests

Type Counter

Access R

Description The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity.

snmpOutGetNexts

Type Counter

Access R

Description The total number of SNMP Get-Next PDUs which have been generated by the SNMP protocol entity.

snmpOutSetRequests

Type Counter

Access R

Description The total number of SNMP Set-Request PDUs which have been generated by the SNMP protocol entity.

snmpOutGetResponses

Type Counter

Access R

Description The total number of SNMP Get-Response PDUs which have been generated by the SNMP protocol entity.

snmpOutTraps

Type Counter

Access R

Description The total number of SNMP Trap PDUs which have been generated by the SNMP protocol entity.

snmpEnableAuthenTraps

Type INTEGER

Access RW

Description Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information; as such, it provides a means whereby all authentication-failure traps may be disabled. Note that it is strongly recommended that this object be stored in non-volatile memory so that it remains constant between re-initializations of the network management system.

Values:

enabled(1),

disabled(2)

Fabric Element Management MIB

There are five groups of objects defined in the Fabric Element Management MIB.

Fabric Element Management MIB Tables

Predefined types

DisplayString

Syntax OCTET STRING

MilliSeconds

Syntax INTEGER (0..2147383647)

2^31 – 1

MicroSeconds

Syntax INTEGER (0..2147383647)

FcNameId

Syntax OCTET STRING (SIZE(8))

Description World wide Name or Fibre Channel Name associated with an FC entity. It's a Network_Destination_ID or Network_Source_ID composed of a value up to 60 bits wide, occupying the remaining 8 bytes while the first nibble identifies the format of the Name_Identifier with hex values: 0: ignored, 1: IEEE 48-bit address, 2: IEEE extended, 3: Locally assigned, 4: 32-bit IP address.

FabricName

Syntax FcNameId

Description The Name Identifier of a Fabric. Each Fabric shall provide a unique Fabric Name. Only the following formats are allowed: IEEE48, and Local.

FcPortName

Syntax FcNameId

Description The Name Identifier associated with a port. Only the following formats are allowed: IEEE48, IEEE extended, and Local.

FcAddressId

Syntax OCTET STRING (SIZE (3))

Description Fibre Channel Address Identifier. A 24-bit value unique within the address space of a Fabric.

FcRxDataFieldSize

Syntax INTEGER (128..2112)

Description Receive Data_Field Size.

FcBbCredit

Syntax INTEGER (0..32767)

Description Buffer-to-buffer Credit.

FcphVersion

Syntax INTEGER (0..255)

Description

FcStackedConnMode

Syntax INTEGER

Description The values are defined as follow: none(1), transparent(2), lockedDown(3).

FcCosCap

Syntax INTEGER (0..127)

Description bit 0 Class F

bit 1 Class 1

bit 2 Class 2
bit 3 Class 3
bit 4 Class 4
bit 5 Class 5
bit 6 Class 6
bit 7 reserved for future

Fc0BaudRate

Syntax INTEGER

Description The values are defined as follow:

other(1)	none of below
oneEighth(2)	155 Mbaud (12.5MB/s)
quarter(4)	266 Mbaud (25.0MB/s)
half(8)	532 Mbaud (50.0MB/s)
full(16)	1 Gbaud (100MB/s)
double(32)	2 Gbaud (200MB/s)
quadruple(64)	4 Gbaud (400MB/s)

Fc0BaudRateCap

Syntax INTEGER (0..127)

Description

bit 0	other
bit 1	oneEighth
bit 2	quarter
bit 3	half
bit 4	full
bit 5	double
bit 6	quadruple
bit 7	reserved for future

Fc0MediaCap

Syntax INTEGER (0..65535)

Description

bit 0	unknown
bit 1	single mode fibre (sm)
bit 2	multi-mode fibre 50 micron (m5)
bit 3	multi-mode fibre 62.5 micron (m6)
bit 4	video cable (tv)
bit 5	miniature cable (mi)
bit 6	shielded twisted pair (stp)
bit 7	twisted wire (tw)
bit 8	long video (lv)
bits 9-15	reserved for future use.

Fc0Medium

Syntax INTEGER

Description The values are defined as follows:

unknown(1)

sm(2)

m5(4)

m6(8)

tv(16)

mi(32)

stp(64)

tw(128)

lv(256)

Fc0TxType

Syntax INTEGER

Description The values are defined as follows:

unknown(1)	
longWaveLaser(2)	(LL)
shortWaveLaser(3)	(SL)
longWaveLED(4)	(LE)
electrical(5)	(EL)
shortWaveLaser-noOFC(6)	(SN)

Fc0Distance

Syntax INTEGER

Description The values are defined as follow: unknown(1), long(2), intermediate(3), short(4).

FcFeModuleCapacity

Syntax INTEGER (1..256)

Description

FcFeFxPortCapacity

Syntax INTEGER (1..256)

Description

FcFeModuleIndex

Syntax INTEGER (1..256)

Description

FcFeFxPortIndex

Syntax INTEGER (1..256)

FcFeNxPortIndex

Syntax INTEGER (1..126)

FcFxPortMode

Syntax	INTEGER
Description	The values are defined as follow: unknown(1), fPort(2), flPort(3).

FcBbCreditModel

Syntax	INTEGER
Description	The values are defined as follow: regular(1), alternate(2).

MIB objects defined in the Fabric Element MIB:

fcFabricName

Type	FabricName
Access	R
Description	The Name_Identifier of the Fabric to which this Fabric Element belongs.

FcElementName

Type	FcNameId
Access	R
Description	The Name_Identifier of the Fabric Element.

FcFeModuleCapacity

Type	FcFeModuleCapacity
Access	R
Description	The maximum number of modules in the Fabric Element, regardless of their current state.

Module Table	A table that contains one entry for each module in the Fabric Element, containing information about the modules
	Fabric Element MIB Object Name

fcFeModuleDescr

Type	DisplayString (SIZE(256))
Provided By	McK DEV_TBL
Access	R
Description	A textual description of the module. This value should include the full name and version identification of the module. It should contain printable ASCII characters. This string should be derived from VPD information stored in the FRU EEPROM.

FcFeModuleObjectID

Type	OBJECT IDENTIFIER
Provided By	SNMP
Access	R
Description	This is a fixed object identifier assigned from the McDATA enterprise subtree (1.3.6.1.4.1.289.2.1.1.2).

fcFeModuleOperStatus

Type	INTEGER
Provided By	SNMP
Access	R
Description	This object indicates the operational status of the module: online(1) – the module is functioning properly; offline(2) – the module is not available; testing(3) – the module is under testing; and faulty(4) – the module is defective in some way. The status is evaluated from `fcFPortPhysOperStatus` as following order. Testing(3): the module is under testing if all four ports on the current module are testing. faulty(4): the module is defective if any of the ports on the current module is faulty. Online(1): the module is functioning properly if any of the ports on the current module is online or testing.

offline(2): the module is not available if any of the ports on the current module is offline.

FcFeModuleLastChange

Type	TIMETICKS
Provided By	SNMP
Access	R
Description	This object contains the value of sysUpTime when the module entered its current operational status. A value of zero indicates that the operational status of the module has not changed since the agent last restarted. This is SS_TIM_RD_TICKS(MILLISEC) * 10.

fcFeModuleFxPortCapacity

Type	FcFeFxPortCapacity
Provided By	AS
Access	R
Description	The number of Fx_Port that can be contained within the module. Within each module, the ports are uniquely numbered in the range from 1 to *fcFeModuleFxPortCapacity* inclusive. However, the numbers are not required to be contiguous. This is AS_glob.prod_cnfg_ptr->ports_per_module.

fcFeModuleName

Type	FcNameId
Provided By	PCP
Access	R
Description	The Name_Identifier of the module. This is the port module world wide name.

Fx_Port Configuration Table

A table that contains one entry for each Fx_Port in the Fabric Element, containing configuration and service parameters of the Fx_Ports.

fcFxConfFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized. This number ranges from 1 to AS_glob.prod_cnfg_ptr->ports_per_module.

FcFxPortName

Type	FcPortName
Provided By	PCP
Access	R
Description	The name identifier of this Fx_Port. Each Fx_Port has a unique port name within the address space of the Fabric. This is the WWN assigned to the port.

FcFxPortFcphVersionHigh

Type	FcphVersion
Provided By	FC2
Access	R
Description	The highest or most recent version of FC-PH that the Fx_Port is configured to support. Since the switch is not capable of changing its support for FC-PH version, the version reported is the one currently in use for this port. If there is no device logged in, then the value is 0. If a device is logged in, the values reported are: 6 = FC-PH 4.0 7 = FC-PH 4.1 8 = FC-PH 4.2 9 = FC-PH 4.3

0x10 = FC-PH2

0x20 = FC-PH3

FcFxPortFcphVersionLow

Type	FcphVersion
Provided By	FC2
Access	R
Description	The lowest or earliest version of FC-PH that the Fx_Port is configured to support. Since the switch is not capable of changing its support for FC-PH version, the version reported is the one currently in use for this port. If there is no device logged in, then the value is 0. For values see *FcFxPortFcphVersionHigh* on page 2-61.

FcFxPortBbCredit

Type	FcBbCredit
Provided By	PCP
Access	R
Description	The total number of receive buffers available for holding Class 1 connect-request, Class 2 or 3 frames from the attached Nx_Port. It is for buffer-to-buffer flow control in the direction from the attached Nx_Port (if applicable) to F_port.

FcFxPortRxBufSize

Type	FcRxDataFieldSize
Provided By	LOGIN SERVER
Access	R
Description	The largest Data_Field Size (in octets) for an FT_1 frame that can be received by the Fx_Port. This is fixed at 2112.

FcFxPortRatov

Type	MilliSeconds

Provided By	PCP
Access	R
Description	The Resource_Allocation_Timeout Value configured for the Fx_Port. This is used as the timeout value for determining when to reuse an NxPort resource such as a Recovery_Qualifier. It represents E_D_TOV (see next object) plus twice the maximum time that a frame may be delayed within the Fabric and still be delivered.

FcFxPortEdtov

Type	MilliSeconds
Provided By	PCP
Access	R
Description	The E_D_TOV value configured for the Fx_Port. The Error_Detect_Timeout Value is used as the timeout value for detecting an error condition.

FcFxPortCosSupported

Type	FcCosCap
Provided By	SNMP
Access	R
Description	A value indicating the set of Classes of Service supported by the Fx_Port. This is fixed at CLASS_2 | CLASS_3 (0x0C).

fcFxPortIntermixSupported

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not the Fx_Port supports an Intermixed Dedicated Connection. The values are defined as follow: yes(1) and no(2). This is fixed at no(2).

FcFxPortStackedConnMode

Type	FcStackedConnMode
Provided By	SNMP
Access	R
Description	A value indicating the mode of Stacked Connect supported by the Fx_Port. This is fixed at none(1).

FcFxPortClass2SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not Class 2 Sequential Delivery is supported by the Fx_Port. The values are defined as follow: yes(1) and no(2). This is fixed at yes(1).

FcFxPortClass3SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not Class 3 Sequential Delivery is supported by the Fx_Port. The values are defined as follow: yes(1) and no(2). This is fixed at yes(1).

FcFxPortHoldTime

Type	MicroSeconds
Provided By	PCP
Access	R

Description The maximum time (in microseconds) that the Fx_Port shall hold a frame before discarding the frame if it is unable to deliver the frame. The value 0 means that the Fx_Port does not support this parameter.

This is equal to quarter of d the E_D_TOV which is obtained from PCP.

FcFxPortBaudRate

Type Fc0BaudRate

Provided By FPM

Access R

Description The FC-0 baud rate of the Fx_Port.

One of these values, or no value will be returned.

0x10, 1 Gbaud (100 MB/s)

0x20, 2 Gbaud (200 MB/s)

0x40 4 Gbaud (400 MB/s)

FcFxPortMedium

Type Fc0Medium

Provided By FPM

Access R

Description The FC-0 medium of the Fx_Port.

The value is a bitwise OR of these values:

0x02 Single Mode fibre

0x04 Multi-mode fibre 50 micron

0x08 Multi-mode fibre 62.5 micron

Or it will be unknown (0x01) if no information is available.

FcFxPortTxType

Type Fc0TxType

Provided By FPM

Access R

Description The FC-0 transmitter type of the Fx_Port.

1 Unknown (long distance laser)

2 LongwaveLaser (LC version)

3 ShortwaveLaser

6 ShortwaveLaser-no OFC

FcFxPortDistance

Type Fc0Distance

Provided By FPM

Access R

Description The FC-0 distance range of the Fx_Port transmitter.

1 Unknown

2 Long

3 Intermediate

4 Short

Fx_Port Operation Table

A table that contains, one entry for each Fx_Port in the Fabric Element, operational status and parameters of the Fx_Ports.

---

fcFxPortOperFxPortIndex

Type FcFeFxPortIndex

Provided By SNMP

Access R

Description This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortID

Type FcAddressId

Provided By Login Server

Access R

Description The address identifier by which this Fx_Port is identified within the Fabric. The Fx_Port may assign its address identifier to its attached NxPort(s) during Fabric Login.

Return a port id if the port is logged into the fabric, otherwise this address is 000000 in FCEOS.

fcFPortAttachedPortName

Type FcPortName

Provided By Login Server

Access R

Description The port name of the attached N_Port, if applicable. If the value of this object is '0000000000000000'H, this F_Port has no NxPort attached to it. This variable has been deprecated and may be implemented for backward compatibility. Not supported for NL_ports.

FcFPortConnectedPort

Type FcAddressId

Provided By SNMP

Access R

Description The address identifier of the destination Fx_Port with which this Fx_Port is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this Fx_Port is not engaged in a class 1 connection. This variable has been deprecated and may be implemented for backward compatibility.

This address is fixed at 0x000000.

FcFxPortBbCreditAvailable

Type Gauge

Provided By PSCC

Access R

Description The number of buffers currently available for receiving frames from the attached port in the buffer-to-buffer flow control. The value should be less than or equal to `fcFxPortBbCredit`.

FcFxPortOperMode

Type FcFxPortMode

Provided By AS

Access R

Description The current operational mode of the Fx_Port.

This value is F_Port(2) if the port_state_data is unavailable or the port is an F_Port, or unknown(1) for the other port state.

FcFxPortAdminMode

Type FcFxPortMode

Provided By AS

Access R

Description The desired operational mode of the Fx_Port.

This value is F_Port(2) if the port_state_data is unavailable or the port is an F_Port, or unknown(1) for the other port state.

Fx_Port Physical Level Table

A table that contains one entry for each Fx_Port in the Fabric Element, containing physical level status and parameters of the Fx_Ports

---

fcFxPortPhysFxPortIndex

Type FcFeFxPortIndex

Provided By SNMP

Access R

Description This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortPhysAdminStatus

Type	INTEGER
Provided By	PCP, FPM
Access	R/W
Description	The desired state of the Fx_Port. A management station may place the Fx_Port in a desired state by setting this object accordingly. The testing(3) state indicates that no operational frames can be passed. When a Fabric Element initializes, all Fx_Port start with `fcFxPortPhysAdminStatus` in the offline(2) state. As the result of either explicit management action or per configuration information accessible by the Fabric Element, `fcFxPortPhysAdminStatus` is then changed to either the online(1) or testing(3) states, or remains in the offline state. The values are defined as follow: online(1) – place port online, offline(2) – take port offline, testing (3). If the port cannot be set to testing because it is inactive or in a failed state, the return value will be resource_unavailable(13).

FcFxPortPhysOperStatus

Type	INTEGER
Provided By	FPM, SNMP
Access	R
Description	The current operational status of the Fx_Port. The testing(3) status indicates that no operational frames can be passed. If *FcFxPortPhysAdminStatus* is offline(2) then *FcFxPortPhysOperStatus* should be offline(2). If *FcFxPortPhysAdminStatus* is changed to online(1) then *FcFxPortPhysOperStatus* should change to online(1) if the Fx_Port is ready to accept Fabric Login request from the attached NxPort; it should proceed and remain in the link-failure(4) state if and only if there is a fault that prevents it from going to the online(1) state. The values are defined as online(1) – Login may proceed, offline(2) – Login cannot proceed, testing(3) – port is under test, link-failure(4) – failure after online/testing.

FcFxPortPhysLastChange

Type	TimeTicks
Provided By	SNMP
Access	R
Description	The value of sysUpTime at the time the Fx_Port entered its current operational status. A value of zero indicates that the Fx_Port's operational status has not changed since the agent last restarted. This is SS_TIM_RD_TICKS(MILLISEC) * 10.

FcFxPortPhysRttov

Type	MilliSeconds
Provided By	SNMP
Access	R
Description	The Receiver_Transmitter_Timeout value of the Fx_Port. This is used by the receiver logic to detect Loss of Synchronization. This value is fixed at 100ms.

Fx_Port Fabric Login Table

An entry containing service parameters established from a successful Fabric Login.

fcFxlogiFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxlogiNxPortIndex

Type	FcFeNxPortIndex

Provided By SNMP

Access R

Description The object identifies the associated NxPort in the attachment for which the entry contains information.

FcFxPortFcphVersionAgreed

Type FcphVersion

Provided By Login Server

Access R

Description The version of FC-PH that the Fx_Port has agreed to support from the Fabric Login.

FcFxPortNxPortBbCredit

Type FcBbCredit

Provided By Login Server

Access R

Description The total number of buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames to be transmitted to the attached NxPort. It is for buffer- to-buffer flow control in the direction from Fx_Port to Nx_Port. The buffer-to-buffer flow control mechanism is indicated in the respective `fcFxPortBbCreditModel`.

FcFxPortNxPortRxDataFieldSize

Type FcRxDataFieldSize

Provided By Login Server

Access R

Description The Receive Data Field Size of the attached NxPort. This is a binary value that specifies the largest Data Field Size for an FT_1 frame that can be received by the NxPort. The value is in number of bytes and ranges from 128 to 2112 inclusive.

FcFxPortCosSuppAgreed

Type FcCosCap

Provided By	Login Server
Access	R
Description	A variable indicating that the attached NxPort has requested the Fx_Port for the support of classes of services and the Fx_Port has granted the request.

FcFxPortIntermixSuppAgreed

Type	INTEGER
Provided By	SNMP
Access	R
Description	A variable indicating that the attached Nx_Port has requested the Fx_Port for the support of Intermix and the Fx_Port has granted the request. This flag is only valid if Class 1 service is supported. The values are defined as yes(1) and no(2). This is always no(2).

FcFxPortStackedConnModeAgreed

Type	FcStackedConnMode
Provided By	SNMP
Access	R
Description	A variable indicating whether the Fx_Port has agreed to support stacked connect from the Fabric Login. This is only meaningful if Class 1 service has been agreed. This is always none(1).

FcFxPortClass2SeqDelivAgreed

Type	INTEGER
Provided By	Login Server
Access	R
Description	A variable indicating whether the Fx_Port has agreed to support Class 2 sequential delivery from the Fabric Login. This is only meaningful if Class 2 service has been agreed. The values are defined as yes(1) and no(2).

FcFxPortClass3SeqDelivAgreed

Type	INTEGER
Provided By	Login Server
Access	R
Description	A flag indicating whether the Fx_Port has agreed to support Class 3 sequential delivery from the Fabric Login. This is only meaningful if Class 3 service has been agreed. The values are defined as yes(1) and no(2).

FcFxPortNxPortName

Type	FcPortName
Provided By	Login Server
Access	R

Description — The port name of the attached Nx_Port, if applicable. If the value of this object is '0000000000000000'H, this Fx_Port has no Nx_Port attached to it.

This is the world wide Name of the attached Nx_Port. It's same as *fcFPortAttachedPortName*.

FcFxPortConnectedNxPort

Type — FcAddressId

Provided By — SNMP

Access — R

Description — The address identifier of the destination Fx_Port with which this Fx_Port is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this Fx_Port is not engaged in a connection.

This is fixed at '000000'H.

fcFxPortBbCreditModel

Type — FcBbCreditModel

Provided By — SNMP

Access — R

Description — This object identifies the BB_Credit model used by the Fx_Port. The regular model refers to the Buffer-to-Buffer flow control mechanism defined in FC-PH [1] is used between the F_Port and the N_Port. For FL_Ports, the Alternate Buffer-to-Buffer flow control mechanism as defined in FC-AL [4] is used between the FL_Port and any attached NL_Ports.

This is fixed at regular(1).

Fx_Port Error Table A table that contains one entry for each Fx_Port, counters that record the numbers of errors detected.

---

fcFxPortErrorFxPortIndex

Type FcFeFxPortIndex

Provided By SNMP

Access R

Description This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortLinkFailures

Type Counter

Provided By PSCC

Access R

Description The number of link failures detected by this Fx_Port.

FcFxPortSyncLosses

Type Counter

Provided By PSCC

Access R

Description The number of loss of synchronization detected by the Fx_Port.

FcFxPortSigLosses

Type Counter

Provided By PSCC

Access R

Description The number of loss of signal detected by the Fx_Port.

FcFxPortPrimSeqProtoErrors

Type	Counter
Provided By	PSCC
Access	R
Description	The number of primitive sequence protocol errors detected by the Fx_Port.

FcFxPortInvalidTxWords

Type	Counter
Provided By	PSCC
Access	R
Description	The number of invalid transmission word detected by the Fx_Port.

FcFxPortInvalidCrcs

Type	Counter
Provided By	PSCC
Access	R
Description	The number of invalid CRC detected by the Fx_Port.

FcFxPortDelimiterErrors

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Delimiter Errors detected by this Fx_Port.

FcFxPortAddressIdErrors

Type	Counter
Provided By	PSCC
Access	R
Description	The number of address identifier errors detected by this Fx_Port.

FcFxPortLinkResetIns

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Link Reset Protocol received by this Fx_Port from the attached Nx_Port.

FcFxPortLinkResetOuts

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Link Reset Protocol issued by this Fx_Port to the attached Nx_Port.

FcFxPortOlsIns

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Offline Sequence received by this Fx_Port.

FcFxPortOlsOuts

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Offline Sequence issued by this Fx_Port.

Class 1 Accounting table

A table that contains one entry for each Fx_Port in the Fabric Element, Class 1 accounting information. These entries are all zero excep for the index, since class 1 is not supported.

fcFxPortC1AcctFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

FcFxPortC1InConnections

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 connections successfully established in which the attached Nx_Port is the source of the connect-request. This value is fixed at 0.

FcFxPortC1OutConnections

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 connections successfully established in which the attached Nx_Port is the destination of the connect-request. This value is fixed at 0.

FcFxPortC1FbsyFrames

Type	Counter
Provided By	SNMP
Access	R
Description	The number of F_BSY frames generated by this Fx_Port against Class 1 connect-request. This value is fixed at 0.

FcFxPortC1FrjtFrames

Type	Counter
Provided By	SNMP
Access	R
Description	The number of F_RJT frames generated by this Fx_Port against Class 1 connect-request. This value is fixed at 0.

FcFxPortC1ConnTime

Type	Counter
Provided By	SNMP
Access	R
Description	The cumulative time that this Fx_Port has been engaged in Class 1 connection. The amount of time of each connection is counted in octets from after a connect- request has been accepted until the connection is disengaged, either by an EOFdt or Link Reset. This value is fixed at 0.

FcFxPortC1InFrames

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 frames (other than Class 1 connect-request) received by this Fx_Port from its attached Nx_Port. This value is fixed at 0.

FcFxPortC1OutFrames

Type	Counter
Provided By	SNMP
Access	R

Description — The number of Class 1 frames (other than Class 1 connect-request) delivered through this Fx_Port to its attached Nx_Port.

This value is fixed at 0.

FcFxPortC1InOctets

Type — Counter

Provided By — SNMP

Access — R

Description — The number of Class 1 frame octets, including the frame delimiters, received by this Fx_Port from its attached Nx_Port.

This value is fixed at 0.

FcFxPortC1OutOctets

Type — Counter

Provided By — SNMP

Access — R

Description — The number of Class 1 frame octets, including the frame delimiters, delivered through this Fx_Port its attached Nx_Port.

This value is fixed at 0.

FcFxPortC1Discards

Type — Counter

Provided By — SNMP

Access — R

Description — The number of Class 1 frames discarded by this Fx_Port.

This value is fixed at 0.

Class 2 Accounting table

A table that contains one entry for each Fx_Port in the Fabric Element, Class 2 accounting information recorded since the management agent has re-initialized.

fcFxPortC2AcctFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortC2InFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frames received by this Fx_Port from its attached Nx_Port.

FcFxPortC2OutFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frames delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC2InOctets

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frame octets, including the frame delimiters, received by this Fx_Port from its attached Nx_Port.

FcFxPortC2OutOctets

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frame octets, including the frame delimiters, delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC2Discards

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 2 frames discarded by this Fx_Port. This value is not supported. It's always zero.

FcFxPortC2FbsyFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of F_BSY frames generated by this Fx_Port against Class 2 frames.

FcFxPortC2FrjtFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of F_RJT frames generated by this Fx_Port against Class 2 frames.

Class 3 Accounting table

A table that contains one entry for each Fx_Port in the Fabric Element, Class 3 accounting information recorded since the management agent has re-initialized.

fcFxPortC3AcctFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortC3InFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frames received by this Fx_Port from its attached Nx_Port.

FcFxPortC3OutFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frames delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC3InOctets

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frame octets, including the frame delimiters, received by this Fx_Port from its attached Nx_Port.

FcFxPortC3OutOctets

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frame octets, including the frame delimiters, delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC3Discards

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frames discarded by this Fx_Port.

Fx_Port Capability Table

A table that contains one entry for each Fx_Port, the capabilities of the port within the Fabric Element

fcFxPortCapFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortCapFcphVersionHigh

Type	FcphVersion
Provided By	FC2
Access	R

Description The highest or most recent version of FC-PH that the Fx_Port is capable of supporting. For values see *FcFxPortFcphVersionHigh* on page 2-61.

FcFxPortCapFcphVersionLow

Type FcphVersion

Provided By FC2

Access R

Description The lowest or earliest version of FC-PH that the Fx_Port is capable of supporting. For values see *FcFxPortFcphVersionHigh* on page 2-61.

FcFxPortCapBbCreditMax

Type FcBbCredit

Provided By SNMP

Access R

Description The maximum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached Nx_Port.

This value is fixed at 16.

FcFxPortCapBbCreditMin

Type FcBbCredit

Provided By SNMP

Access R

Description The minimum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached Nx_Port.

This value is fixed at 1.

FcFxPortCapRxDataFieldSizeMax

Type FcRxDataFieldSize

Provided By SNMP

Access R

Description The maximum size in bytes of the Data Field in a frame that the Fx_Port is capable of receiving from its attached Nx_Port.

This value is fixed at 2112.

FcFxPortCapRxDataFieldSizeMin

Type FcRxDataFieldSize

Provided By SNMP

Access R

Description The minimum size in bytes of the Data Field in a frame that the Fx_Port is capable of receiving from its attached Nx_Port.

This value is fixed at 2112.

FcFxPortCapCos

Type FcCosCap

Provided By SNMP

Access R

Description A value indicating the set of Classes of Service that the Fx_Port is capable of supporting.

This value is fixed at CLASS_2 | CLASS_3 (0x0C).

---

fcFxPortCapIntermix

Type INTEGER

Provided By SNMP

Access R

Description A flag indicating whether or not the Fx_Port is capable of supporting the intermixing of Class 2 and Class 3 frames during a Class 1 connection. This flag is only valid if the port is capable of supporting Class 1 service. The values are defined as follow: yes(1) and no(2).

This value is fixed no(2).

FcFxPortCapStackedConnMode

Type FcStackedConnMode

Provided By	SNMP
Access	R
Description	A value indicating the mode of Stacked Connect request that the Fx_Port is capable of supporting. This value is fixed at none(1).

FcFxPortCapClass2SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not the Fx_Port is capable of supporting Class 2 Sequential Delivery. This value is fixed at yes(1).

FcFxPortCapClass3SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not the Fx_Port is capable of supporting Class 3 Sequential Delivery. This value is fixed at yes(1).

FcFxPortCapHoldTimeMax

Type	MicroSeconds
Provided By	SNMP
Access	R
Description	The maximum holding time (in microseconds) that the Fx_Port is capable of supporting. This value is not supported. It's always zero.

FcFxPortCapHoldTimeMin

Type	MicroSeconds
Provided By	SNMP
Access	R
Description	The minimum holding time (in microseconds) that the Fx_Port is capable of supporting. This value is not supported. It's always zero.

FcFxPortCapBaudRates

Type	Fc0BaudRateCap
Provided By	FPM
Access	R
Description	A value indicating the set of baud rates that the Fx_Port is capable of supporting. This variable has been deprecated and may be implemented for backward compatibility.

FcFxPortCapMedia

Type	Fc0MediaCap
Provided By	FPM
Access	R
Description	A value indicating the set of media that the Fx_Port is capable of supporting.

NOTE: All the counters are 32-bit counters.

FCMGMT-MIB

```
Last edit date: Aug 19th, 2001
DEFINITIONS ::= BEGIN
IMPORTS
IpAddress, TimeTicks, experimental
        FROM RFC1155-SMI
        FROM RFC-1212
DisplayString
        FROM RFC1213-MIB
TRAP-TYPE
        FROM RFC-1215;
Textual conventions for this MIB
FcNameId ::= OCTET STRING (SIZE(8))
FcGlobalId ::= OCTET STRING (SIZE(16))
FcAddressId ::= OCTET STRING (SIZE(3))
FcEventSeverity ::= INTEGER {
unknown (1),
emergency (2),
alert  (3),
critical (4),
error (5),
warning (6),
notify (7),
info (8),
debug (9),
mark (10)All messages logged
}
```

```
FcUnitType ::= INTEGER {
unknown(1),
other(2),                    none of the following
hub(3),                      passive connectivity unit supporting loop
                             protocol.
switch(4),                   active connectivity unit supporting
                             multiple protocols.
gateway(5),                  unit that converts not only the interface but
                             also encapsulates the frame into another
                             protocol. The assumption is that there is
                             always two gateways connected together.
                             For example, FC <-> ATM.
converter(6),                unit that converts from one interface to
                             another. For example, FC <-> SCSI.
hba(7),                      host bus adapter
proxy-agent(8),              software proxy-agent
storage-device(9),           disk,cd,tape,etc
host(10),                    host computer
storage-subsystem(11),       raid, library, etc
module(12),                  subcomponent of a system
swdriver(13),                software driver
storage-access-device(14),   Provides storage management and access
                             for hetergeneous hosts and heterogeneous
                             devices.
wdm(15),                     waveform division
mutiplexer ups(16),          uninterruptable power supply
nas(17)                      network attached storage (NFS/CIFS)

}
fcmgmt    OBJECT IDENTIFIER ::= { experimental 94 }
```

Groups in fcmgmt

```
connSet                 OBJECT IDENTIFIER ::= { fcmgmt 1 }
trapReg                 OBJECT IDENTIFIER ::= { fcmgmt 2 }
statSet                 OBJECT IDENTIFIER ::= { fcmgmt 4 }
```

connUnitServiceSet	OBJECT IDENTIFIER ::= { fcmgmt 5 }
connUnitServiceScalars	OBJECT IDENTIFIER ::= { connUnitServiceSet 1 }
connUnitServiceTables	OBJECT IDENTIFIER ::= { connUnitServiceSet 2 }

revisionNumber

Syntax: DisplayString (SIZE (4))

Access: read-only

Status: mandatory

Description: This is the revision number for this MIB. The format of the revision value is as follows

(0) = high order major revision number

(1) = low order major revision number

(2) = high order minor revision number

(3) = low order minor revision number

The value will be stored as an ASCII value. The following is the current value of this object.

(0) = '0'

(1) = '4'

(2) = '0'

(3) = '0'

This defines a revision of 04.00

Sequence: `::= { fcmgmt 3 }`

Connectivity Unit Group

Implementation of the group is mandatory for all systems.

uNumber

Syntax: INTEGER

Access: read-only

Status: mandatory

Description The number of connectivity units present on this system (represented by this agent). May be a count of the boards in a chassis or the number of full boxes in a rack.

DEFVAL {1}

Sequence `::= {congest 1}`

---

systemURL

Syntax DisplayString

Access read-write

Status mandatory

Description The top-level URL of the system. If it does not exist the value is empty string. The URL format is implementation dependant and can have keywords embedded that are preceeded by a percent sign (eg, %USER). The following are the defined keywords that will be recognized and replaced with data during a launch.

USER	replace with username
PASSWORD	replace with password
GLOBALID	replace with globalid
SERIALNO	replace with serial number

A management application will read this object from the MIB, provide values for any of the keywords listed above that are present in the string, and then use the URL to invoke or launch the program referenced. If write is not supported, then return invalid.

This value will be retained across boots.

DEFVAL { “” }

Sequence `::= { connSet 2 }`

---

statusChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the last status change occurred for any members of the set.

Sequence ::= { connSet 3 }

configurationChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the last configuration change occurred for any members of the set. This represents a union of change information for connUnitConfigurationChangeTime.

Sequence ::= { connSet 4 }

connUnitTableChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the connUnitTable was updated (an entry was either added or deleted.

Sequence ::= { connSet 5 }

The Connectivity table contains general information on the system's units.

connUnitTable

Syntax SEQUENCE OF ConnUnitEntry

Access not-accessible

Status mandatory

Description The connectivity table contains general information on the system's units. This is the list of units under a single SNMP agent. The number of entries is given by the value of uNumber. It is 1 for stand-alone system.

Sequence ::= { connSet 6 }

connUnitEntry

Syntax	ConnUnitEntry
Access	not-accessible
Status	mandatory
Description	A connectivity unit entry containing objects for a particular unit.

```
INDEX { connUnitId }
::= { connUnitTable 1 }
ConnUnitEntry ::=
SEQUENCE {
connUnitId
        FcGlobalId,
connUnitGlobalId
        FcGlobalId,
connUnitType
        FcUnitType,
connUnitNumports
        INTEGER,
connUnitState
        INTEGER,
connUnitStatus
        INTEGER,
connUnitProduct
        DisplayString,
connUnitSn
        DisplayString,
connUnitUpTime
        TimeTicks,
connUnitUrl
        DisplayString,
```

connUnitDomainId
OCTET STRING,
connUnitProxyMaster
INTEGER,
connUnitPrincipal
INTEGER,
connUnitNumSensors
INTEGER,
connUnitStatusChangeTime
TimeTicks,
connUnitConfigurationChangeTime
TimeTicks,
connUnitNumRevs
INTEGER,
connUnitNumZones
INTEGER,
connUnitModuleId
FcGlobalId,
connUnitName
DisplayString,
connUnitInfo
DisplayString,
connUnitControl
INTEGER,
connUnitContact
DisplayString,
connUnitLocation
DisplayString,
connUnitEventFilter

FcEventSeverity,
connUnitNumEvents
INTEGER,
connUnitMaxEvents
INTEGER,
connUnitEventCurrID
INTEGER,
connUnitFabricID
FcGlobalId,
connUnitNumLinks
INTEGER,
connUnitVendorId
DisplayString }

---

connUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The unique identification for this connectivity unit among those within this proxy domain. The value must be unique within the proxy domain because it is the index variable for connUnitTable. The value assigned to a given conectivity unit should be persistent across agent and unit resets. It should be the same as connUnitGlobalId if connUnitGlobalId is known and stable.
Sequence	`::= { connUnitEntry 1 }`

---

connUnitGlobalId

Syntax	FcGlobalId
Access	read-only
Status	mandatory

Description

An optional global-scope identifier for this connectivity unit. It MUST be a WWN for this connectivity unit or 16 octets of value zero.

WWN formats requiring fewer than 16 octets MUST be extended to 16 octets with trailing zero octets, Left justified, zero filled, If a WWN is used for connUnitId, the same WWN MUST be used for connUnitGlobalId. When a non-zero value is provided, it SHOULD be persistent across agent and unit resets. It SHOULD be globally unique.

It SHOULD be one of these FC-PH/PH3 formats:

IEEE (NAA=1)

IEEE Extended (NAA=2)

IEEE Registered (NAA=5).

IEEE Registered extended (NAA=6).

Use of the IEEE formats allows any IEEE-registered vendor to assure global uniqueness independently.

The following are some references on IEEE WWN formats:

http://standards.ieee.org/regauth/oui/tutorials/fibreformat.html

http://standards.ieee.org/regauth/oui/tutorials/fibrecomp_id.html

If one or more WWNs are associated with the connUnit via other management methods, one of them SHOULD be used for connUnitGlobalId. If there is not a WWN assigned specifically to the connUnit, there is some merit, though not a requirement, to using a WWN assigned to (one of) its permanently attached FC/LAN interface(s). This can not risk uniqueness, though. As a counterexample, if your agent runs in a host and the host has an HBA, it is quite possible that agent, host, and HBA will all be distinct connUnits, so the host and agent can not use the WWN of the HBA.

Another example:

If your hub has a built-in Ethernet port, it might be reasonable for the hub to use its MAC address (prefixed with the appropriate NAA) as its connUnitId. But if the Ethernet were a replaceable PCCard, the hub should have an independent ID."

Sequence

```
::= { connUnitEntry 2 }
```

connUnitType

Syntax	FcUnitType
Access	read-only
Status	mandatory
Description	The type of this connectivity unit.
Sequence	`::= { connUnitEntry 3 }`

connUnitNumports

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of physical ports in the connectivity unit (internal/embedded, external).
Sequence	`::= { connUnitEntry 4 }`

connUnitState

Syntax	INTEGER {	
	unknown(1),	
	online(2),	available for meaningful work
	offline(3)	unavailable for meaningful work, for example in self-test mode, configuration, etc.
	}	
Access	read-only	
Status	mandatory	
Description	This object reports the overall state of the connectivity unit. The meaning of all values is essentially self-explanatory. Any of these values may occur with any of the ConnUnitStatus values.	
Sequence	`::= { connUnitEntry 5 }`	

connUnitStatus

Syntax	INTEGER {
	unknown(1),
	unused(2), cannot report status
	ok(3), available for meaningful work
	warning(4), something needs attention
	failed(5) something has failed
	}
Access	read-only
Status	mandatory
Description	Overall status of the connectivity unit. The goal of this object is to be the single poll point to check the status of the connunit. If there is any other component that has warning, then this should be set to warning, etc. Any of these values may occur with any of the ConnUnitState values.
Sequence	`::= { connUnitEntry 6 }`

connUnitProduct

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The connectivity unit vendor's product model name.
Sequence	`::= { connUnitEntry 7 }`

connUnitSn

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The serial number for this connectivity unit.
Sequence	`::= { connUnitEntry 8 }`

connUnitUpTime

Syntax	TimeTicks
Access	read-only
Status	mandatory
Description	The number of centiseconds since the last unit initialization.
Sequence	`::= { connUnitEntry 9 }`

connUnitUrl

Syntax	DisplayString
Access	read-write
Status	mandatory
Description	URL to launch a management application, if applicable. Otherwise empty string. In a standalone unit, this would be the same as the top-level URL. This has the same definition as systemURL for keywords. If write is not supported, then return invalid. This value will be retained across boots.
Sequence	`::= { connUnitEntry 10 }`

connUnitDomainId

Syntax	OCTET STRING (SIZE(3))
Access	read-only
Status	mandatory
Description	24 bit Fibre Channel address ID of this connectivity unit, right justified with leading zero's if required. This should be set to the Fibre Channel address ID or if it is a switch it would be set to the Domain Controller address. If this value is not applicable, return all bits set to 1.
Sequence	`::= { connUnitEntry 11 }`

connUnitProxyMaster

Syntax	INTEGER {

unknown(1),

no(2),

yes(3)

}

Access read-only

Status mandatory

Description A value of 'yes' means this is the proxy master unit for a set of managed units. For example, this could be the only unit with a management card in it for a set of units. A standalone unit should return 'yes' for this object.

Sequence `::= { connUnitEntry 12 }`

---

connUnitPrincipal

Syntax INTEGER {

unknown(1),

no(2),

yes(3)

}

Access read-only

Status mandatory

Description Whether this connectivity unit is the principal unit within the group of fabric elements. If this value is not applicable, return unknown.

Sequence `::= { connUnitEntry 13 }`

---

connUnitNumSensors

Syntax INTEGER

Access read-only

Status mandatory

Description Number of sensors in the connUnitSensorTable.

Sequence `::= { connUnitEntry 14 }`

connUnitStatusChangeTime

Syntax	TimeTicks
Access	read-only
Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the last status change occurred.
Sequence	`::= { connUnitEntry 15 }`

connUnitConfigurationChangeTime

Syntax	TimeTicks
Access	read-only
Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the last configuration change occurred.
Sequence	`::= { connUnitEntry 16 }`

connUnitNumRevs

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The number of revisions in the connUnitRevsTable. DEFVAL { 1 }
Sequence	`::= { connUnitEntry 17 }`

connUnitNumZones

Syntax	INTEGER
Access	read-only
Status	obsolete
Description	Number of zones defined in connUnitZoneTable.
Sequence	`::= { connUnitEntry 18 }`

connUnitModuleId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	This is a unique id, persistent between boots, that can be used to group a set of connUnits together into a module. The intended use would be to create a connUnit with a connUnitType of 'module' to represent a physical or logical group of connectivity units. Then the value of the group would be set to the value of connUnitId for this 'container' connUnit. connUnitModuleId should be zeros if this connUnit is not part of a module.
Sequence	`::= { connUnitEntry 19 }`

connUnitName

Syntax	DisplayString (SIZE(0..79))
Access	read-write
Status	mandatory
Description	A display string containing a name for this connectivity unit. This object value should be persistent between boots.
Sequence	`::= { connUnitEntry 20 }`

connUnitInfo

Syntax	DisplayString
Access	read-write
Status	mandatory
Description	A display string containing information about this connectivity unit. This object value should be persistent between boots.
Sequence	`::= { connUnitEntry 21 }`

connUnitControl

Syntax	INTEGER{
	unknown(1),

invalid(2),

resetConnUnitColdStart(3),

resetConnUnitWarmStart(4),

offlineConnUnit(5),

onlineConnUnit(6)

}

Access read-write

Status mandatory

Description This object is used to control the addressed connUnit.

NOTE: 'Cold Start' and 'Warm Start' are as defined in MIB II and are not meant to be a factory reset.

`resetConnUnitColdStart`: the addressed unit performs a 'Cold Start' reset.

`resetConnUnitWarmStart`: the addressed unit performs a 'Warm Start' reset.

`offlineConnUnit`: the addressed unit puts itself into an implementation dependant 'offline' state. In general, if a unit is in an offline state, it cannot be used to perform meaningful Fibre Channel work.

`onlineConnUnit`: the addressed unit puts itself into an implementation dependant 'online' state. In general, if a unit is in an online state, it is capable of performing meaningful Fibre Channel work.

NOTE: Each implementation may chose not to allow any or all of these values on a SET.

Sequence `::= { connUnitEntry 22 }`

connUnitContact

Syntax DisplayString (SIZE (0..79))

Access read-write

Status mandatory

Description Contact information for this connectivity unit. Persistent across boots.

Sequence `::= { connUnitEntry 23 }`

connUnitLocation

Syntax DisplayString (SIZE (0..79))

Access read-write

Status mandatory

Description Location information for this connectivity unit.Persistent across boots.

Sequence `::= { connUnitEntry 24 }`

connUnitEventFilter

Syntax FcEventSeverity

Access read-write

Status mandatory

Description This value defines the event severity that will be logged by this connectivity unit. All events of severity less than or equal to connUnitEventFilter are logged in connUnitEventTable. Persistent across boots.

Sequence `::= { connUnitEntry 25 }`

connUnitNumEvents

Syntax INTEGER

Access read-only

Status mandatory

Description Number of events currently in the connUnitEventTable.

Sequence `::= { connUnitEntry 26 }`

connUnitMaxEvents

Syntax INTEGER

Access read-only

Status mandatory

Description Max number of events that can be defined in connUnitEventTable.

Sequence `::= { connUnitEntry 27 }`

---

connUnitEventCurrID

Syntax INTEGER

Access read-only

Status mandatory

Description The last used event id (connUnitEventIndex).

Sequence `::= { connUnitEntry 28 }`

---

connUnitFabricID

Syntax FcGlobalId

Access read-only

Status mandatory

Description A globally unique value to identify the fabric that this ConnUnit belongs to, otherwise empty string. This would typically be equal to the connUnitGlobalID of the primary switch in a Fibre Channel fabric.

Sequence `::= { connUnitEntry 29 }`

---

connUnitNumLinks

Syntax INTEGER

Access read-only

Status mandatory

Description The number of links in the link table.

Sequence `::= { connUnitEntry 30 }`

---

connUnitVendorId

Syntax DisplayString (SIZE (0..79))

Access read-only

Status mandatory

Description The connectivity unit vendor's name.

Sequence `::= { connUnitEntry 31 }`

The Table of revisions for hardware and software elements.

---

connUnitRevsTable

Syntax SEQUENCE OF ConnUnitRevsEntry

Access not-accessible

Status mandatory

Description Table of the revisions supported by connectivity units managed by this agent.

Sequence `::= { connSet 7 }`

---

connUnitRevsEntry

Syntax ConnUnitRevsEntry

Access not-accessible

Status mandatory

Description ""

INDEX { connUnitRevsUnitId,

connUnitRevsIndex }

Sequence `::= { connUnitRevsTable 1 }`

ConnUnitRevsEntry ::=

SEQUENCE {

connUnitRevsUnitId

FcGlobalId,

connUnitRevsIndex

INTEGER,

connUnitRevsRevId

DisplayString,

connUnitRevsDescription

DisplayString }

connUnitRevsUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this revision table.
Sequence	`::= { connUnitRevsEntry 1 }`

connUnitRevsIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitRevsEntrys with the same value of connUnitRevsUnitId, in the range between 1 and connUnitNumRevs[connUnitRevsUnitId].
Sequence	`::= { connUnitRevsEntry 2 }`

connUnitRevsRevId

Syntax	DisplayString
	Access read-only
Status	mandatory
Description	A vendor-specific string identifying a revision of a component of the connUnit indexed by connUnitRevsUnitId.
Sequence	`::= { connUnitRevsEntry 3 }`

connUnitRevsDescription

Syntax	DisplayString
Access	read-only
Status	mandatory

Description — Description of a component to which the revision corresponds.

Sequence — `::= { connUnitRevsEntry 4 }`

The Sensor table

connUnitSensorTable

Syntax — SEQUENCE OF ConnUnitSensorEntry

Access — not-accessible

Status — mandatory

Description — Table of the sensors supported by each connectivity unit managed by this agent.

Sequence — `::= { connSet 8 }`

connUnitSensorEntry

Syntax — ConnUnitSensorEntry

Access — not-accessible

Status — mandatory

Description — Each entry contains the information for a specific sensor.

INDEX { connUnitSensorUnitId,

connUnitSensorIndex }

Sequence — `::= { connUnitSensorTable 1 }`

ConnUnitSensorEntry ::=

SEQUENCE {

connUnitSensorUnitId

    FcGlobalId,

connUnitSensorIndex

    INTEGER (1..2147483647),

connUnitSensorName

    DisplayString,

connUnitSensorStatus

INTEGER,
connUnitSensorInfo
DisplayString,
connUnitSensorMessage
DisplayString,
connUnitSensorType
INTEGER,
connUnitSensorCharacteristic
INTEGER }

---

connUnitSensorUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this sensor table.
Sequence	`::= { connUnitSensorEntry 1 }`

---

connUnitSensorIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitSensorEntrys with the same value of connUnitSensorUnitId, in the range between 1 and connUnitNumSensor[connUnitSensorUnitId].
Sequence	`::= { connUnitSensorEntry 2}`

---

connUnitSensorName

Syntax	DisplayString
Access	read-only
Status	mandatory

Description A textual identification of the sensor intended primarily for operator use.

Sequence `::= { connUnitSensorEntry 3 }`

---

connUnitSensorStatus

Syntax INTEGER{

unknown(1),	
other(2),	the sensor indicates other than ok, warning or failure.
ok(3),	the sensor indicates ok
warning(4),	the sensor indicates a warning
failed(5)	the sensor indicates failure

}

Access read-only

Status mandatory

Description The status indicated by the sensor.

Sequence `::= { connUnitSensorEntry 4 }`

---

connUnitSensorInfo

Syntax DisplayString

Access read-only

Status mandatory

Description Miscellaneous static info about the sensor such as its serial number.

Sequence `::= { connUnitSensorEntry 5 }`

---

connUnitSensorMessage

Syntax DisplayString

Access read-only

Status mandatory

Description This describes the status of the sensor as a message. It may also provide more resolution on the sensor indication, for example 'Cover temperature 1503K, above nominal operating range'

Sequence `::= { connUnitSensorEntry 6 }`

---

connUnitSensorType

Syntax INTEGER{
unknown(1),
other(2),
battery(3),
fan(4),
power-supply(5),
transmitter(6),
enclosure(7),
board(8),
receiver(9)
}

Access read-only

Status mandatory

Description The type of component being monitored by this sensor.

Sequence `::= { connUnitSensorEntry 7 }`

---

connUnitSensorCharacteristic

Syntax INTEGER{
unknown(1),
other(2),
temperature(3),
pressure(4),
emf(5),
currentValue(6), -- current is a keyword

airflow(7),
frequency(8),
power(9),
door(10)
}

Access read-only
Status mandatory
Description The characteristics being monitored by this sensor.
Sequence `::= { connUnitSensorEntry 8 }`

The port table

connUnitPortTable

Syntax SEQUENCE OF ConnUnitPortEntry
Access not-accessible
Status mandatory
Description Generic information on ports for a specific connUnit.
Sequence `::= { connSet 10 }`

connUnitPortEntry

Syntax ConnUnitPortEntry
Access not-accessible
Status mandatory
Description Each entry contains the information for a specific port.
INDEX { connUnitPortUnitId,
connUnitPortIndex }
Sequence `::= { connUnitPortTable 1 }`
ConnUnitPortEntry ::=
SEQUENCE {

connUnitPortUnitId
FcGlobalId,
connUnitPortIndex
INTEGER,
connUnitPortType
INTEGER,
connUnitPortFCClassCap
OCTET STRING,
connUnitPortFCClassOp
OCTET STRING,
connUnitPortState
INTEGER,
connUnitPortStatus
INTEGER,
connUnitPortTransmitterType
INTEGER,
connUnitPortModuleType
INTEGER,
connUnitPortWwn
FcGlobalId,
connUnitPortFCId
FcAddressId,
connUnitPortSn
DisplayString,
connUnitPortRevision
DisplayString,
connUnitPortVendor
DisplayString,
connUnitPortSpeed

INTEGER,
connUnitPortControl
INTEGER,
connUnitPortName
DisplayString,
connUnitPortPhysicalNumber
INTEGER,
connUnitPortStatObject
OBJECT IDENTIFIER,
connUnitPortProtocolCap
OCTET STRING,
connUnitPortProtocolOp
OCTET STRING,
connUnitPortNodeWwn
FcNameId,
connUnitPortHWState
INTEGER }

connUnitPortUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this port.
Sequence	`::= { connUnitPortEntry 1 }`

connUnitPortIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory

Description A unique value among all connUnitPortEntrys on this connectivity unit, between 1 and connUnitNumPort[connUnitPortUnitId].

Sequence `::= { connUnitPortEntry 2 }`

---

connUnitPortType

Syntax INTEGER{

unknown (1),	
other (2),	
not-present (3),	
hub-port (4),	
n-port (5),	end port for fabric
nl-port (6),	end port for loop
fl-port (7),	public loop
f-port (8),	fabric port
e-port (9),	fabric expansion port
g-port (10),	generic fabric port
domain-ctl (11),	domain controller
hub-controller(12),	
scsi (13),	parallel SCSI port
escon (14),	
lan (15),	
wan (16),	
ac (17),	AC power line
dc (18),	DC power line
ssa (19),	serial storage architecture
wdm (20),	optical wave division multiplex
ib (21),	Infiniband
ipstore (22)	IP storage

}

Access read-only

Status mandatory

Description The port type.

Sequence `::= { connUnitPortEntry 3 }`

connUnitPortFCClassCap

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the classes of service capability of this port. If this is not applicable, return all bits set to zero. The bits have the following definition:

unknown	0
class-f	1
class-one	2
class-two	4
class-three	8
class-four	16
class-five	32
class-six	64

Sequence `::= { connUnitPortEntry 4 }`

connUnitPortFCClassOp

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the classes of service that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortFCClassCap.

Sequence `::= { connUnitPortEntry 5 }`

connUnitPortState

Syntax	INTEGER{	
	unknown(1),	
	online(2),	available for meaningful work
	offline(3),	not available for meaningful work
	bypassed(4),	no longer used (4/12/00)
	diagnostics(5)	
	}	
Access	read-only	
Status	mandatory	
Description	The user selected state of the port hardware.	
Sequence	`::= { connUnitPortEntry 6 }`	

connUnitPortStatus

Syntax	INTEGER{	
	unknown (1),	
	unused (2),	device cannot report this status
	ready (3),	FCAL Loop or FCPH Link reset protocol initialization has completed
	warning (4),	do not use (4/12/00)
	failure (5),	do not use (4/12/00)
	notparticipating (6),	loop notparticipating and does not have a loop address
	initializing (7),	protocol is proceeding
	bypass (8),	do not use (4/12/00)
	ols (9),	FCP offline status
	other (10)	status not described above
	}	
Access	read-only	
Status	mandatory	

Description An overall protocol status for the port. This value of connUnitPortState is not online, then this is reported Unknown.

Sequence `::= { connUnitPortEntry 7 }`

connUnitPortTransmitterType

Syntax INTEGER{
unknown(1),
other(2),
unused(3),
shortwave(4),
longwave(5),
copper(6),
scsi(7),
longwaveNoOFC(8),
shortwaveNoOFC(9),
longwaveLED(10),
ssa(11)}

Access read-only

Status mandatory

Description The technology of the port transceiver.

Sequence `::= { connUnitPortEntry 8 }`

connUnitPortModuleType

Syntax INTEGER{
unknown(1),
other(2),
gbic(3),
embedded(4), -- fixed, ie, oneXnine
glm(5),
gbicSerialId(6),

gbicNoSerialId(7),

gbicNotInstalled(8),

smallFormFactor(9) -- this is generically a small form factor connector.

}

Access read-only

Status mandatory

Description The module type of the port connector.

Sequence `::= { connUnitPortEntry 9 }`

---

connUnitPortWwn

Syntax FcGlobalId

Access read-only

Status mandatory

Description The World Wide Name of the port if applicable, otherwise all zeros.

Sequence `::= { connUnitPortEntry 10 }`

---

connUnitPortFCId

Syntax FcAddressId

Access read-only

Status mandatory

Description This is the assigned Fibre Channel ID of this port. This value is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected. If this is an eport, then it will only contain the domain ID left justified, zero filled. If this port does not have a Fibre Channel address, return all bits set to 1.

Sequence `::= { connUnitPortEntry 11 }`

---

connUnitPortSn

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The serial number of the unit (e.g., for a GBIC). If this is not applicable, return empty string.

Sequence `::= { connUnitPortEntry 12 }`

connUnitPortRevision

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The port revision (e.g., for a GBIC).

Sequence `::= { connUnitPortEntry 13 }`

connUnitPortVendor

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The port vendor (e.g., for a GBIC).

Sequence `::= { connUnitPortEntry 14 }`

connUnitPortSpeed

Syntax INTEGER

Access read-only

Status mandatory

Description The speed of the port in kilobytes per second.

Sequence `::= { connUnitPortEntry 15 }`

connUnitPortControl

Syntax INTEGER{

unknown(1),

invalid(2),

resetConnUnitPort(3),

bypassConnUnitPort(4),

unbypassConnUnitPort(5),

offlineConnUnitPort(6),

onlineConnUnitPort(7),

resetConnUnitPortCounters(8)

}

Access read-write

Status mandatory

Description This object is used to control the addressed connUnit's port. Valid commands are:

`resetConnUnitPort`: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'reset' operation. Examples of these operations are: the Link Reset protocol, the Loop Initialization protocol, or a resynchronization occurring between the transceiver in the addressed port to the transceiver that the port is connected to.

`bypassConnUnitPort`: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'bypass' operation. Examples of these operations are: transitioning from online to offline, a request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.

`unbypassConnUnitPort`: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'unbypass' operation. Examples of these operations are: the Link Failure protocol, a request (PARTICIPATING) command to the Loop Port state machine, or addition of the port to an arbitrated loop by a hub.

`offlineConnUnitPort`: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'offline' operation. Examples of these operations are: disabling a port's transceiver, the Link Failure protocol, request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.

`onlineConnUnitPort`: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'online' operation. Examples of these operations are: enabling a port's transceiver, the Link Failure protocol, request (PARTICIPATING) command to the Loop Port state machine, or addition of the port from an arbitrated loop by a hub.

`resetConnUnitPortCounters`: If the addressed connUnit allows this operation to be performed to this port, the addressed port statistics table counters will be set to zero.

NOTE: Each implementation may chose not to allow any or all of these values on a SET. On a read, if you do not support write, then return invalid. Otherwise return the last control operation attempted.

Sequence `::= { connUnitPortEntry 16 }`

connUnitPortName

Syntax DisplayString

Access read-write

Status mandatory

Description A user-defined name for this port. This means that up to DisplayString characters may be supported. If less than, then the name will be truncated in the connunit.

Sequence `::= { connUnitPortEntry 17 }`

connUnitPortPhysicalNumber

Syntax INTEGER

Access read-only

Status mandatory

Description This is the internal port number this port is known by. In many implementations, this should be the same as connUnitPortIndex. Some implementations may have an internal port representation not compatible with the rules for table indices. In that case, provide the internal representation of this port in this object. This value may also be used in the `connUnitLinkPortNumberX` or `connUnitLinkPortNumberY` objects of the connUnitLinkTable.

Sequence `::= { connUnitPortEntry 18 }`

---

connUnitPortStatObject

Syntax OBJECT IDENTIFIER

Access read-only

Status deprecated

Description This contains the OID of the first object of the table that contains the statistics for this particular port. If this has a value of zero, then there are no statistics available for this port. The port type information will help identify the statistics objects that will be found in the table.

Sequence `::= { connUnitPortEntry 19 }`

---

connUnitPortProtocolCap

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the driver level protocol capability of this port. If this is not applicable, return all bits set to zero. The bits have the following definition:

unknown	- 0
Loop	1
Fabric	2
SCSI	4
TCP/IP	8
VI	16
FICON	32

Sequence `::= { connUnitPortEntry 20 }`

---

connUnitPortProtocolOp

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the driver level protocol(s) that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortProtocolCap.

Sequence `::= { connUnitPortEntry 21 }`

connUnitPortNodeWwn

Syntax FcNameId

Access read-only

Status mandatory

Description The Node World Wide Name of the port if applicable, otherwise all zeros. This should have the same value for a group of related ports. The container is defined as the largest physical unit.

For example, all ports on HBAs on a host will have the same Node WWN. All ports on the same storage subsystem will have the same Node WWN.

Sequence `::= { connUnitPortEntry 22 }`

connUnitPortHWState

Syntax INTEGER{

unknown (1),

failed (2), port failed diagnostics

bypassed (3), FCAL bypass, loop only

active (4), connected to a device

loopback (5), Port in ext loopback

txfault (6), Transmitter fault

noMedia (7), media not installed

linkDown (8) waiting for activity (rx sync)

}

Access read-only

Status mandatory

Description The hardware detected state of the port.

Sequence `::= { connUnitPortEntry 23 }`

Event group

connUnitEventTable

Syntax	SEQUENCE OF ConnUnitEventEntry
Access	not-accessible
Status	mandatory
Description	The table of connectivity unit events. Errors, warnings, and information should be reported in this table.
Sequence	`::= { connSet 11 }`

connUnitEventEntry

Syntax	ConnUnitEventEntry
Access	not-accessible
Status	mandatory
Description	Each entry contains information on a specific event for the given connectivity unit. INDEX { connUnitEventUnitId, connUnitEventIndex }

Sequence	`::= { connUnitEventTable 1 }`
	ConnUnitEventEntry ::=
	SEQUENCE {
	connUnitEventUnitId
	FcGlobalId,
	connUnitEventIndex
	INTEGER (1..2147483647),
	connUnitEventId
	INTEGER,
	connUnitREventTime
	DisplayString,
	connUnitSEventTime
	TimeTicks,

connUnitEventSeverity
FcEventSeverity,
connUnitEventType
INTEGER,
connUnitEventObject
OBJECT IDENTIFIER,
connUnitEventDescr
DisplayString
}

connUnitEventUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this event table.
Sequence	`::= { connUnitEventEntry 1 }`

connUnitEventIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Each connectivity unit has its own event buffer. s it wraps, it may write over previous events.This object is an index into the buffer. It is recommended that this table be read using 'getNext's to retrieve the initial table. The management application should read the event table at periodic intervals and then determine if any new entries were added by comparing the last known index value with the current highest index value. The management application should then update its copy of the event table. If the read interval is too long, it is possible that there may be events that may not be contained in the agent's internal event buffer.

For example, an agent may read events 50-75. At the next read interval, connUnitEventCurrID is 189. If the management app tries to read event index 76, and the agent's internal buffer is 100 entries max, event index 76 will no longer be available.

The index value is an incrementing integer starting from one every time there is a table reset. On table reset, all contents are emptied and all indeces are set to zero. When an event is added to the table, the event is assigned the next higher integer value than the last item entered into the table. If the index value reaches its maximum value, the next item entered will cause the index value to roll over and start at one again.

Sequence `::= { connUnitEventEntry 2 }`

---

connUnitEventId

Syntax INTEGER

Access read-only

Status deprecated

Description The internal event Id. Incremented for each event, ranging between 1 and connUnitMaxEvents.

Not used as table index to simplify the agent implementation. When this reaches the end of the range specified by connUnitMaxEvents, the Id will roll over to start at one. This value will be set back to one at reset. The relationship of this value to the index is that internal event id may represent a smaller number than a 32 bit integer (eg max 100 entries) and would only have a value range up to connUnitMaxEvents.

Sequence `::= { connUnitEventEntry 3 }`

---

connUnitREventTime

Syntax DisplayString (SIZE (0..15))

Access read-only

Status mandatory

Description This is the real time when the event occurred. It has the following format.

DDMMYYYY HHMMSS

DD=day number
MM=month number
YYYY=year number
HH=hour number
MM=minute number
SS=seconds number
If not applicable, return either a NULL string or '00000000 000000'.

Sequence `::= { connUnitEventEntry 4 }`

connUnitSEventTime

Syntax TimeTicks
Access read-only
Status mandatory
Description This is the sysuptime timestamp when the event occurred.
Sequence `::= { connUnitEventEntry 5 }`

connUnitEventSeverity

Syntax FcEventSeverity
Access read-only
Status mandatory
Description The event severity level.
Sequence `::= { connUnitEventEntry 6 }`

connUnitEventType

Syntax INTEGER{
unknown(1),
other(2),
status(3),
configuration(4),
topology(5)

}

Access read-only

Status mandatory

Description The type of this event.

Sequence `::= { connUnitEventEntry 7 }`

---

connUnitEventObject

Syntax OBJECT IDENTIFIER

Access read-only

Status mandatory

Description This is used with the connUnitEventType to identify which object the event refers to.

Examples are connUnitPortStatus.connUnitId.connUnitPortIndex, connUnitStatus.connUnitId, etc.

Sequence `::= { connUnitEventEntry 8 }`

---

connUnitEventDescr

Syntax DisplayString

Access read-only

Status mandatory

Description The description of the event.

Sequence `::= { connUnitEventEntry 9 }`

The link table is intended to organize and communicate any information the agent possesses which would assist a management application to discover the ONNECTIVITY UNITS in the framework and the TOPOLOGY of their interconnect. That is, the goal is to assist the management application not only to LIST the elements of the framework, but to MAP them. With this goal, the agent SHOULD include as much as it possesses about any links from its own connectivity units to others, including links among its own units. An agent SHOULD include partial information about links if it is not able to fully define them.

For an entry to be considered to be valid, both the X (local) and the Y (remote) need to have one valid value. If the agent is able to discover links which do not directly attach to members of its agency and its discovery algorithm gives some assurance the links are recently valid, it MAY include these links. Link information entered by administrative action MAY be included even if not validated directly if the link has at least one endpoint in this agency, but SHOULD NOT be included otherwise.

A connectivity unit should fill the table in as best it can. One of the methods to fill this in would be to use the RNID ELS (reference ANSI T11.3 draft standard FC-FS). This allows one to query a port for the information needed for the link table.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry

created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table.

connUnitLinkTable

Syntax	SEQUENCE OF ConnUnitLinkEntry
Access	not-accessible
Status	mandatory
Description	A list of links know to this agent from this connectivity unit to other connectivity units.
Sequence	`::= { connSet 12 }`

connUnitLinkEntry

Syntax	ConnUnitLinkEntry
Access	not-accessible
Status	mandatory
Description	An entry describing a particular link to another. INDEX { connUnitLinkUnitId, connUnitLinkIndex }

Sequence `::= { connUnitLinkTable 1 }`

```
ConnUnitLinkEntry ::=
SEQUENCE {
connUnitLinkUnitId
        FcGlobalId,
connUnitLinkIndex
        INTEGER,
connUnitLinkNodeIdX
        OCTET STRING,
connUnitLinkPortNumberX
        INTEGER,
connUnitLinkPortWwnX
        FcGlobalId,
connUnitLinkNodeIdY
        OCTET STRING,
connUnitLinkPortNumberY
        INTEGER,
connUnitLinkPortWwnY
        FcGlobalId,
connUnitLinkAgentAddressY
        OCTET STRING,
connUnitLinkAgentAddressTypeY
        INTEGER,
connUnitLinkAgentPortY
        INTEGER,
connUnitLinkUnitTypeY
        FcUnitType,
connUnitLinkConnIdY
        OCTET STRING,
```

connUnitLinkCurrIndex

INTEGER

}

connUnitLinkUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this link table.
Sequence	`::= { connUnitLinkEntry 1 }`

connUnitLinkIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	This index is used to create a unique value for each entry in the link table with the same connUnitLinkUnitId. The value can only be reused if it is not currently in use and the value is the next candidate to be used. This value wraps at the highest value represented by the size of INTEGER. This value is reset to zero when the system is reset and the first value to be used is one.
Sequence	`::= { connUnitLinkEntry 2 }`

connUnitLinkNodeIdX

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The node WWN of the unit at one end of the link. If the node WWN is unknown and the node is a connUnit in the responding agent then the value of this object MUST BE equal to its connUnitID.
Sequence	`::= { connUnitLinkEntry 3 }`

connUnitLinkPortNumberX

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The port number on the unit specified by connUnitLinkNodeIdX if known, otherwise -1. If the value is nonnegative then it will be equal to connUnitPortPhysicalNumber.
Sequence	`::= { connUnitLinkEntry 4 }`

connUnitLinkPortWwnX

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The port WWN of the unit specified by connUnitLinkNodeIdX if known, otherwise 16 octets of binary 0
Sequence	`::= { connUnitLinkEntry 5 }`

connUnitLinkNodeIdY

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The node WWN of the unit at the other end of the link. If the node WWN is unknown and the node is a connUnit in the responding SNMP agency then the value of this object MUST BE equal to its connUnitID.
Sequence	`::= { connUnitLinkEntry 6 }`

connUnitLinkPortNumberY

Syntax	INTEGER
Access	read-only
Status	mandatory

Description The port number on the unit specified by connUnitLinkNodeIdY if known, otherwise -1. If the value is nonnegative then it will be equal to connUnitPortPhysicalNumber.

Sequence `::= { connUnitLinkEntry 7 }`

---

connUnitLinkPortWwnY

Syntax FcGlobalId

Access read-only

Status mandatory

Description The port WWN on the unit specified by connUnitLinkNodeIdY if known, otherwise 16 octets of binary 0.

Sequence `::= { connUnitLinkEntry 8 }`

---

connUnitLinkAgentAddressY

Syntax OCTET STRING (SIZE(16))

Access read-only

Status mandatory

Description The address of an FCMGMT MIB agent for the node identified by connUnitLinkNodeIdY, if known; otherwise 16 octets of binary 0.

Sequence `::= { connUnitLinkEntry 9 }`

---

connUnitLinkAgentAddressTypeY

Syntax INTEGER

Access read-only

Status mandatory

Description If connUnitLinkAgentAddressY is nonzero, it is a protocol address. ConnUnitLinkAgentAddressTypeY is the the 'address family number' assigned by IANA to identify the address format. (eg, 1 is Ipv4, 2 is Ipv6). If connUnitLinkAgentAddressY is all zeros, then this value is ignored.

Sequence `::= { connUnitLinkEntry 10 }`

connUnitLinkAgentPortY

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The IP port number for the agent. This is provided in case the agent is at a non-standard SNMP port.
Sequence	`::= { connUnitLinkEntry 11 }`

connUnitLinkUnitTypeY

Syntax	FcUnitType
Access	read-only
Status	mandatory
Description	Type of the FC connectivity unit as defined in connUnitType.
Sequence	`::= { connUnitLinkEntry 12 }`

connUnitLinkConnIdY

Syntax	OCTET STRING (SIZE(3))
Access	read-only
Status	mandatory
Description	This is the Fibre Channel ID of this port. If the connectivity unit is a switch, this is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected. If this is an eport, then it will only contain the domain ID. If not any of those, unknown or cascaded loop, return all bits set to 1.
Sequence	`::= { connUnitLinkEntry 13 }`

connUnitLinkCurrIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The last used link index.

Sequence ::= { connUnitLinkEntry 14 }

The Active Zone table is defined by read-only view of the active zone configuration.

Based on the fabric zone server defined in the FC-GS-3 standard and the enhanced zone server proposed for FC-GS-4.

Allows alias zone members (GS-4 feature) and includes an alias table. Zone table and Alias table are flat, double-indexed MIB tables. Database view of inactive zone sets is not provided. (Note: This could be added by adding a third index (connUnitZoneSetIndex) and an zone set active flag to the connUnitActiveZonesTable table.)

Undefined, vendor-specific bytes have not been included for the bit mask objects connUnitZoneCapabilities and connUnitZoneEnforcementState.

---

connUnitZoneTable

Syntax SEQUENCE OF ConnUnitZoneEntry

Access not-accessible

Status mandatory

Description ""

Sequence `::= { connSet 13 }`

---

connUnitZoneEntry

Syntax ConnUnitZoneEntry

Access not-accessible

Status mandatory

Description ""

INDEX { connUnitZoneIndex,

connUnitZoneMemberIndex }

Sequence `::= { connUnitZoneTable 1 }`

ConnUnitZoneEntry ::=

SEQUENCE {

connUnitZoneIndex

INTEGER,
connUnitZoneMemberIndex
INTEGER,
connUnitZoneSetName
DisplayString,
connUnitZoneSetNumZones
INTEGER,
connUnitZoneName
DisplayString,
connUnitZoneCapabilities
OCTET STRING,
connUnitZoneEnforcementState
OCTET STRING,
connUnitZoneAttributeBlock
OCTET STRING,
connUnitZoneNumMembers
INTEGER,
connUnitZoneMemberIdType
INTEGER,
connUnitZoneMemberID
FcGlobalId }

connUnitZoneIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Unique table index for each zone. Valid values are between 1 and connUnitZoneSetNumZones.
Sequence	`::= { connUnitZoneEntry 1 }`

connUnitZoneMemberIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Unique table index for each zone member. Valid values are between 1 and connUnitZoneNumMembers.
Sequence	`::= { connUnitZoneEntry 2 }`

connUnitZoneSetName

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	Name of the active zone set to which the zone and zone member belong.
Sequence	`::= { connUnitZoneEntry 3 }`

connUnitZoneSetNumZones

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The number of zones in the active zone set.
Sequence	`::= { connUnitZoneEntry 4 }`

connUnitZoneName

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	Name of the zone.
Sequence	`::= { connUnitZoneEntry 5 }`

connUnitZoneCapabilities

Syntax	OCTET STRING (SIZE(1))
Access	read-only
Status	mandatory
Description	1-byte bit mask that specifies the zoning capabilities supported by the fabric. Bit 7 - Soft zones supported. Bit 6 - Hard zones supported. Bits 5-0 - Reserved.
Sequence	`::= { connUnitZoneEntry 6 }`

connUnitZoneEnforcementState

Syntax	OCTET STRING (SIZE(1))
Access	read-only
Status	mandatory
Description	1-byte bit mask that specifies the current enforcement of the Zone Set. Bit 7 - Soft zone set enforced. Bit 6 - Hard zone set enforced. Bits 5-0 - Reserved.
Sequence	`::= { connUnitZoneEntry 7 }`

connUnitZoneAttributeBlock

Syntax	OCTET STRING (SIZE(80))
Access	read-only
Status	mandatory
Description	A variable length structure that contains extended zone attributes defined in the FC-GS-4 enhanced zone server. See FC-GS-4 draft standard for details and format of the structure. Support of this object is optional.
Sequence	`::= { connUnitZoneEntry 8 }`

connUnitZoneNumMembers

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of zone members in the zone: connUnitZoneName.
Sequence	`::= { connUnitZoneEntry 9 }`

connUnitZoneMemberIdType

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Type of zone member ID: 1- Port WWN 2- Domain & Port ID 3- FC Address 4- Node WWN 5- Alias Name 6-'FF'h - Vendor specified.
Sequence	`::= { connUnitZoneEntry 10 }`

connUnitZoneMemberID

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	ID of the zone member based on connUnitZoneMemberIdType.
Sequence	`::= { connUnitZoneEntry 11 }`

Zone Alias Table

connUnitZoningAliasTable

Syntax	SEQUENCE OF ConnUnitZoningAliasEntry
Access	not-accessible
Status	mandatory
Description	“”
Sequence	`::= { connSet 14 }`

connUnitZoningAliasEntry

Syntax	ConnUnitZoningAliasEntry
Access	not-accessible
Status	mandatory
Description	“”
	INDEX { connUnitZoningAliasIndex,
	connUnitZoningAliasMemberIndex }
Sequence	`::= { connUnitZoningAliasTable 1 }`
	ConnUnitZoningAliasEntry ::=
	SEQUENCE {
	connUnitZoningAliasIndex
	INTEGER,
	connUnitZoningAliasMemberIndex
	INTEGER,
	connUnitZoningAliasNumAliases
	INTEGER,
	connUnitZoningAliasName
	DisplayString,
	connUnitZoningAliasNumMembers
	INTEGER,

connUnitZoningAliasMemberIdType
INTEGER,
connUnitZoningAliasMemberID
FcGlobalId }

connUnitZoningAliasIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Unique table index for each alias. Valid values are between 1 and connUnitZoningAliasNumAliases.
Sequence	`::= { connUnitZoningAliasEntry 1 }`

connUnitZoningAliasMemberIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Unique table index for each alias member. Valid values are between 1 and connUnitZoningAliasNumMembers.
Sequence	`::= { connUnitZoningAliasEntry 2 }`

connUnitZoningAliasNumAliases

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The number of aliases defined in this table.
Sequence	`::= { connUnitZoningAliasEntry 3 }`

connUnitZoningAliasName

Syntax	DisplayString (SIZE (0..79))

Access	read-only
Status	mandatory
Description	The alias name.
Sequence	`::= { connUnitZoningAliasEntry 4 }`

---

connUnitZoningAliasNumMembers

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of members in the alias: connUnitZoningAliasName.
Sequence	`::= { connUnitZoningAliasEntry 5 }`

---

connUnitZoningAliasMemberIdType

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Type of alias member ID: 1- Port WWN 2- Domain & Port ID 3- FC Address Others: reserved.
Sequence	`::= { connUnitZoningAliasEntry 6 }`

---

connUnitZoningAliasMemberID

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	ID of the alias member based on connUnitZoningAliasMemberIdType.
Sequence	`::= { connUnitZoningAliasEntry 7 }`

The following four tables have been obsoleted. These were used to keep statistic information based on the type of port type. It was changed for all ports to use a common statistics table.

connUnitPortStatHubTable

Syntax	SEQUENCE OF ConnUnitPortStatHubEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the hub port type.
Sequence	`::= { statSet 1 }`

connUnitPortStatFabricTable

Syntax	SEQUENCE OF ConnUnitPortStatFabricEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the fabric port types.
Sequence	`::= { statSet 2 }`

connUnitPortStatSCSITable

Syntax	SEQUENCE OF ConnUnitPortStatSCSIEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the SCSI port type.
Sequence	`::= { statSet 3 }`

connUnitPortStatLANTable

Syntax	SEQUENCE OF ConnUnitPortStatLANEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the LAN/WAN port type.

Sequence `::= { statSet 4 }`

There is one and only one statistics table for each individual port. For all objects in statistics table, if the object is not supported by the conn unit then the high order bit is set to 1 with all other bits set to zero. The high order bit is reserved to indicate if the object if supported or not. All objects start at a value of zero at hardware initialization and continue incrementing till end of 63 bits and then wrap to zero.

Port Statistics

connUnitPortStatTable

Syntax	SEQUENCE OF ConnUnitPortStatEntry
Access	not-accessible
Status	mandatory
Description	A list of statistics for the fabric port types.
Sequence	`::= { statSet 5 }`

connUnitPortStatEntry

Syntax	ConnUnitPortStatEntry
Access	not-accessible
Status	mandatory
Description	An entry describing port statistics. INDEX { connUnitPortStatUnitId, connUnitPortStatIndex }

Sequence	`::= { connUnitPortStatTable 1 }`
	ConnUnitPortStatEntry ::=
	SEQUENCE {
	connUnitPortStatUnitId
	FcGlobalId,
	connUnitPortStatIndex
	INTEGER,
	connUnitPortStatCountError
	OCTET STRING,
	connUnitPortStatCountTxObjects
	OCTET STRING,
	connUnitPortStatCountRxObjects
	OCTET STRING,

connUnitPortStatCountTxElements
OCTET STRING,
connUnitPortStatCountRxElements
OCTET STRING,
connUnitPortStatCountBBCreditZero
OCTET STRING,
connUnitPortStatCountInputBuffersFull
OCTET STRING,
connUnitPortStatCountFBSYFrames
OCTET STRING,
connUnitPortStatCountPBSYFrames
OCTET STRING,
connUnitPortStatCountFRJTFrames
OCTET STRING,
connUnitPortStatCountPRJTFrames
OCTET STRING,
connUnitPortStatCountClass1RxFrames
OCTET STRING,
connUnitPortStatCountClass1TxFrames
OCTET STRING,
connUnitPortStatCountClass1FBSYFrames
OCTET STRING,
connUnitPortStatCountClass1PBSYFrames
OCTET STRING,
connUnitPortStatCountClass1FRJTFrames
OCTET STRING,
connUnitPortStatCountClass1PRJTFrames
OCTET STRING,
connUnitPortStatCountClass2RxFrames

OCTET STRING,
connUnitPortStatCountClass2TxFrames
OCTET STRING,
connUnitPortStatCountClass2FBSYFrames
OCTET STRING,
connUnitPortStatCountClass2PBSYFrames
OCTET STRING,
connUnitPortStatCountClass2FRJTFrames
OCTET STRING,
connUnitPortStatCountClass2PRJTFrames
OCTET STRING,
connUnitPortStatCountClass3RxFrames
OCTET STRING,
connUnitPortStatCountClass3TxFrames
OCTET STRING,
connUnitPortStatCountClass3Discards
OCTET STRING,
connUnitPortStatCountRxMulticastObjects
OCTET STRING,
connUnitPortStatCountTxMulticastObjects
OCTET STRING,
connUnitPortStatCountRxBroadcastObjects
OCTET STRING,
connUnitPortStatCountTxBroadcastObjects
OCTET STRING,
connUnitPortStatCountRxLinkResets
OCTET STRING,
connUnitPortStatCountTxLinkResets
OCTET STRING,

connUnitPortStatCountNumberLinkResets
OCTET STRING,
connUnitPortStatCountRxOfflineSequences
OCTET STRING,
connUnitPortStatCountTxOfflineSequences
OCTET STRING,
connUnitPortStatCountNumberOfflineSequences
OCTET STRING,
connUnitPortStatCountLinkFailures
OCTET STRING,
connUnitPortStatCountInvalidCRC
OCTET STRING,
connUnitPortStatCountInvalidTxWords
OCTET STRING,
connUnitPortStatCountPrimitiveSequenceProtocolErrors
OCTET STRING,
connUnitPortStatCountLossofSignal
OCTET STRING,
connUnitPortStatCountLossofSynchronization
OCTET STRING,
connUnitPortStatCountInvalidOrderedSets
OCTET STRING,
connUnitPortStatCountFramesTooLong
OCTET STRING,
connUnitPortStatCountFramesTruncated
OCTET STRING,
connUnitPortStatCountAddressErrors
OCTET STRING,
connUnitPortStatCountDelimiterErrors

OCTET STRING,

connUnitPortStatCountEncodingDisparityErrors

OCTET STRING

}

connUnitPortStatUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this port stat table.
Sequence	`::= { connUnitPortStatEntry 1 }`

connUnitPortStatIndex

Syntax	INTEGER(0..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all entries in this table having the same connUnitPortStatUnitId, between 1 and connUnitNumPort[connUnitPortStatUnitId].
Sequence	`::= { connUnitPortStatEntry 2 }`

connUnitPortStatCountError

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A count of the errors that have occured on this port.
Sequence	`::= { connUnitPortStatEntry 3 }`

connUnitPortStatCountTxObjects

Syntax	OCTET STRING (SIZE (8))

Access	read-only
Status	mandatory
Description	The number of frames/packets/IOs/etc that have been transmitted by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Tx objects.
Sequence	`::= { connUnitPortStatEntry 4 }`

connUnitPortStatCountRxObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of frames/packets/IOs/etc that have been received by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Rx objects.
Sequence	`::= { connUnitPortStatEntry 5 }`

connUnitPortStatCountTxElements

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of octets or bytes that have been transmitted by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.
Sequence	`::= { connUnitPortStatEntry 6 }`

connUnitPortStatCountRxElements

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description The number of octets or bytes that have been received. by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.

Sequence `::= { connUnitPortStatEntry 7 }`

---

connUnitPortStatCountBBCreditZero

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of transitions in/out of BBcredit zero state. The other side is not providing any credit. Note, this is a Fibre Channel stat only.

Sequence `::= { connUnitPortStatEntry 8 }`

---

connUnitPortStatCountInputBuffersFull

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of occurrences when all input buffers of a port were full and outbound buffer-to-buffer credit transitioned to zero. There is no credit to provide to other side. Note, this is a Fibre Channel stat only.

Sequence `::= { connUnitPortStatEntry 9 }`

---

connUnitPortStatCountFBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. Port can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

Sequence ::= { connUnitPortStatEntry 10 }

---

connUnitPortStatCountPBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that PBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if the destination port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

Sequence ::= { connUnitPortStatEntry 11 }

---

connUnitPortStatCountFRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Frame that was rejected by the fabric. Note, This is the total for all classes and is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 12 }

---

connUnitPortStatCountPRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Frame that was rejected at the destination N_Port. Note, This is the total for all classes and is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 13 }

connUnitPortStatCountClass1RxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 1 Frames received at this port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 14 }`

connUnitPortStatCountClass1TxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 1 Frames transmitted out this port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 15 }`

connUnitPortStatCountClass1FBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 16 }`

connUnitPortStatCountClass1PBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description Count of times that PBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 17 }`

---

connUnitPortStatCountClass1FRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 18 }`

---

connUnitPortStatCountClass1PRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 19 }`

---

connUnitPortStatCountClass2RxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 2 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 20 }`

connUnitPortStatCountClass2TxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 2 Frames transmitted out this port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 21 }`

connUnitPortStatCountClass2FBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 22 }`

connUnitPortStatCountClass2PBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that PBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 23 }`

connUnitPortStatCountClass2FRJTFrames

Syntax	OCTET STRING (SIZE (8))

Access	read-only
Status	mandatory
Description	Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 24 }`

connUnitPortStatCountClass2PRJTFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 25 }`

connUnitPortStatCountClass3RxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 3 Frames received at this port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 26 }`

connUnitPortStatCountClass3TxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 3 Frames transmitted out this port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 27 }`

connUnitPortStatCountClass3Discards

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 3 Frames that were discarded upon reception at this port. There is no FBSY or FRJT generated for Class 3 Frames. They are simply discarded if they cannot be delivered. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 28 }`

connUnitPortStatCountRxMulticastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Multicast Frames or Packets received at this port.
Sequence	`::= { connUnitPortStatEntry 29 }`

connUnitPortStatCountTxMulticastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Multicast Frames or Packets transmitted out this port.
Sequence	`::= { connUnitPortStatEntry 30 }`

connUnitPortStatCountRxBroadcastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Broadcast Frames or Packets received at this port.
Sequence	`::= { connUnitPortStatEntry 31 }`

connUnitPortStatCountTxBroadcastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Broadcast Frames or Packets transmitted out this port. On a Fibre Channel loop, count only OPNr frames generated.
Sequence	`::= { connUnitPortStatEntry 32 }`

connUnitPortStatCountRxLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets. This is the number of LRs received. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 33 }`

connUnitPortStatCountTxLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets. This is the number LRs transmitted. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 34 }`

connUnitPortStatCountNumberLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets and LIPs detected at this port. The number times the reset link protocol is initiated. These are the count of the logical

resets, a count of the number of primatives. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 35 }`

---

connUnitPortStatCountRxOfflineSequences

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Offline Primitive OLS received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 36 }`

---

connUnitPortStatCountTxOfflineSequences

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Offline Primitive OLS transmitted by this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 37 }`

---

connUnitPortStatCountNumberOfflineSequences

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Offline Primitive sequence received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 38 }`

---

connUnitPortStatCountLinkFailures

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of link failures. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 39 }`

---

connUnitPortStatCountInvalidCRC

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of frames received with invalid CRC. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Loop ports should not count CRC errors passing through when monitoring. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 40 }`

---

connUnitPortStatCountInvalidTxWords

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of invalid transmission words received at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 41 }`

---

connUnitPortStatCountPrimitiveSequenceProtocolErrors

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of primitive sequence protocol errors detected at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 42 }`

connUnitPortStatCountLossofSignal

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of instances of signal loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 43 }`

connUnitPortStatCountLossofSynchronization

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of instances of synchronization loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 44 }`

connUnitPortStatCountInvalidOrderedSets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of invalid ordered sets received at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 45 }`

connUnitPortStatCountFramesTooLong

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description — Count of frames received at this port where the frame length was greater than what was agreed to in FLOGI/PLOGI. This could be caused by losing the end of frame delimiter. Note, this is a Fibre Channel only stat.

Sequence — `::= { connUnitPortStatEntry 46 }`

---

connUnitPortStatCountFramesTruncated

Syntax — OCTET STRING (SIZE (8))

Access — read-only

Status — mandatory

Description — Count of frames received at this port where the frame length was less than the minimum indicated by the frame header - normally 24 bytes, but it could be more if the DFCTL field indicates an optional header should have been present. Note, this is a Fibre Channel only stat.

Sequence — `::= { connUnitPortStatEntry 47 }`

---

connUnitPortStatCountAddressErrors

Syntax — OCTET STRING (SIZE (8))

Access — read-only

Status — mandatory

Description — Count of frames received with unknown addressing. e.x. unknown SID or DID. the SID or DID is not known to the routing algorithm. Note. this is a Fibre Channel only stat.

Sequence — `::= { connUnitPortStatEntry 48 }`

---

connUnitPortStatCountDelimiterErrors

Syntax — OCTET STRING (SIZE (8))

Access — read-only

Status — mandatory

Description — Count of invalid frame delimiters received at this port. An example is a frame with a class 2 start and and a class 3 at the end. Note, this is a Fibre Channel only stat.

Sequence — `::= { connUnitPortStatEntry 49 }`

connUnitPortStatCountEncodingDisparityErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of disparity errors received at this port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 50 }`

The Fibre Channel Simple Name Server table

The Fibre Channel Simple Name Server table contains an entry for each device presently known to this connUnit. There will not be any version on this since FC-GS3 does not define a version today.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. The total number of entries are defined by the size of the table.

connUnitSnsMaxEntry

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The current number of entries in the table.
Sequence	`::= { connUnitServiceScalars 1 }`

connUnitSnsTable

Syntax	SEQUENCE OF ConnUnitSnsEntry
Access	not-accessible
Status	mandatory
Description	This table contains an entry for each object registered with this port in the switch.
Sequence	`::= { connUnitServiceTables 1 }`

connUnitSnsEntry

Syntax	ConnUnitSnsEntry
Access	not-accessible
Status	mandatory
Description	The Simple Name Server table for the port represented by ConnUnitSnsPortIndex . INDEX { connUnitSnsId, connUnitSnsPortName,

connUnitSnsPortIdentifier }

Sequence `::= { connUnitSnsTable 1 }`

ConnUnitSnsEntry ::=
SEQUENCE {
connUnitSnsId
OCTET STRING,
connUnitSnsPortIndex
INTEGER,
connUnitSnsPortIdentifier
FcAddressId,
connUnitSnsPortName
FcNameId,
connUnitSnsNodeName
FcNameId,
connUnitSnsClassOfSvc
OCTET STRING,
connUnitSnsNodeIPAddress
OCTET STRING,
connUnitSnsProcAssoc
OCTET STRING,
connUnitSnsFC4Type
OCTET STRING,
connUnitSnsPortType
OCTET STRING,
connUnitSnsPortIPAddress
OCTET STRING,
connUnitSnsFabricPortName
FcNameId,
connUnitSnsHardAddress

```
        FcGlobalId,
connUnitSnsSymbolicPortName
        DisplayString,
connUnitSnsSymbolicNodeName
        DisplayString
        }
```

connUnitSnsId

Syntax	OCTET STRING (SIZE (16))
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this Name Server table.
Sequence	`::= { connUnitSnsEntry 1 }`

connUnitSnsPortIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The physical port number of this SNS table entry. Each physical port has an SNS table with 1-n entries indexed by ConnUnitSnsPortIdentifier (port address)
Sequence	`::= { connUnitSnsEntry 2 }`

connUnitSnsPortIdentifier

Syntax	FcAddressId
Access	read-only
Status	mandatory
Description	The Fibre Channel ID for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 3 }`

connUnitSnsPortName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Port WWN for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 4 }`

connUnitSnsNodeName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Node Name for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 5 }`

connUnitSnsClassOfSvc

Syntax	OCTET STRING (SIZE (1))
Access	read-only
Status	mandatory
Description	The Classes of Service offered by this entry in the SNS table. This is a bit mask where each bit that represents the class of service is set to a value of one if the class is supported. Class 1 is bit zero.
Sequence	`::= { connUnitSnsEntry 6 }`

connUnitSnsNodeIPAddress

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The IPv6 formatted address of the Node for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 7 }`

connUnitSnsProcAssoc

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The Process Associator for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 8 }`

connUnitSnsFC4Type

Syntax	OCTET STRING (SIZE (32))
Access	read-only
Status	mandatory
Description	The FC-4 Types supported by this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 9 }`

connUnitSnsPortType

Syntax	OCTET STRING (SIZE (1))
Access	read-only
Status	mandatory
Description	The Port Type of this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 10 }`

connUnitSnsPortIPAddress

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The IPv6 formatted address of this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 11 }`

connUnitSnsFabricPortName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Fabric Port name of this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 12 }`

connUnitSnsHardAddress

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The Hard ALPA of this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 13 }`

connUnitSnsSymbolicPortName

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The Symbolic Port Name of this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 14 }`

connUnitSnsSymbolicNodeName

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The Symbolic Node Name of this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 15 }`

Platform Table

Design Objectives and Scope

Simple, read-only view of platform registration entries. Platform registry is a service hosted by the connectivity unit, in a very similar manner as the SNS table. I.E., the platform table is contained by the connectivity unit. A platform can register it's attributes and platform nodes with the registry service. (See FC-GS-3 Configuration Server for details.)

The platform table is a flat, double-indexed MIB table. To keep the table simple, only one platform management URL is exposed. If a platform registers more than one management URL, the first one is reported in this table.

This table is based on the fabric configuration server defined in the FC-GS-3 standard and enhanced platform attributes proposed for FC-GS-4. Note that the information contained in this table may only contain the platforms that this connUnit can see or it may contain a fabric wide view of the platforms.

connUnitPlatformMaxEntry

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The maximum number of entries in the platform table.
Sequence	`::= { connUnitServiceScalars 2 }`

connUnitPlatformTable

Syntax	SEQUENCE OF ConnUnitPlatformEntry
Access	not-accessible
Status	mandatory
Description	""
Sequence	`::= { connUnitServiceTables 2 }`

connUnitPlatformEntry

Syntax ConnUnitPlatformEntry

Access not-accessible

Status mandatory

Description ""

INDEX { connUnitPlatformIndex,
connUnitPlatformNodeIndex }

Sequence `::= { connUnitPlatformTable 1 }`

ConnUnitPlatformEntry ::=
SEQUENCE {
connUnitPlatformIndex
INTEGER,
connUnitPlatformNodeIndex
INTEGER,
connUnitPlatformUnitID
FcGlobalId,
connUnitPlatformName
OCTET STRING,
connUnitPlatformType
FcUnitType,
connUnitPlatformLabel
DisplayString,
connUnitPlatformDescription
DisplayString,
connUnitPlatformLocation
DisplayString,
connUnitPlatformManagementUrl
DisplayString,
connUnitPlatformNumNodes

INTEGER,

connUnitPlatformNodeName

FcGlobalId }

---

connUnitPlatformIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Unique table index for each platform. Valid values are between 1 and connUnitPlatformsMaxEntry.
Sequence	`::= { connUnitPlatformEntry 1 }`

---

connUnitPlatformNodeIndex

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	Unique table index for each platform node. Valid values are between 1 and connUnitPlatformsNumNodes.
Sequence	`::= { connUnitPlatformEntry 2 }`

---

connUnitPlatformUnitID

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this Platform table.
Sequence	`::= { connUnitPlatformEntry 3 }`

---

connUnitPlatformName

Syntax	OCTET STRING (SIZE(79))
Access	read-only

Status mandatory

Description The platform name. May be either a readable string or a unique ID format as specified in the FC-GS-4 draft standard.

Sequence `::= { connUnitPlatformEntry 4 }`

connUnitPlatformType

Syntax FcUnitType

Access read-only

Status mandatory

Description The platform type.

Sequence `::= { connUnitPlatformEntry 6 }`

connUnitPlatformLabel

Syntax DisplayString (SIZE (0..79))

Access read-only

Status mandatory

Description An administratively assigned symbolic name for the platform. The Platform Label shall only contain print-able ASCII characters.

Sequence `::= { connUnitPlatformEntry 7 }`

connUnitPlatformDescription

Syntax DisplayString (SIZE (0..79))

Access read-only

Status mandatory

Description A textual description of the platform. This value should include the full name and version identification of the platform's hardware type and software operating system. The Platform Description shall only contain printable ASCII characters.

Sequence `::= { connUnitPlatformEntry 8 }`

connUnitPlatformLocation

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The physical location of the platform (e.g., telephone closet, 3rd floor). The Platform Location shall only contain printable ASCII characters.
Sequence	`::= { connUnitPlatformEntry 9 }`

connUnitPlatformManagementUrl

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	Primary management URL for the platform. If the platform registers more than one URL, then this URL is equal to the first in the list.
Sequence	`::= { connUnitPlatformEntry 10 }`

connUnitPlatformNumNodes

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of nodes contained in the platform.
Sequence	`::= { connUnitPlatformEntry 11 }`

connUnitPlatformNodeName

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	WWN (world wide name) of the node contained by the platform.
Sequence	`::= { connUnitPlatformEntry 12 }`

SNMP trap registration group

trapMaxClients

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The maximum number of SNMP trap recipients supported by the connectivity unit.
Sequence	`::= { trapReg 1 }`

trapClientCount

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The current number of rows in the trap table.
Sequence	`::= { trapReg 2 }`

trapRegTable

Syntax	SEQUENCE OF TrapRegEntry
Access	not-accessible
Status	mandatory
Description	A table containing a row for each IP address/port number that traps will be sent to.
Sequence	`::= { trapReg 3 }`

trapRegEntry

Syntax	TrapRegEntry
Access	not-accessible
Status	mandatory
Description	Ip/Port pair for a specific client.

INDEX {trapRegIpAddress,
trapRegPort }

Sequence `::= { trapRegTable 1 }`

TrapRegEntry ::=
SEQUENCE {
trapRegIpAddress
    IpAddress,
trapRegPort
    INTEGER (1..2147483647),
trapRegFilter
    FcEventSeverity,
trapRegRowState
    INTEGER
    }

---

trapRegIpAddress

Syntax	IpAddress
Access	read-only
Status	mandatory
Description	The Ip address of a client registered for traps.
Sequence	`::= { trapRegEntry 1 }`

---

trapRegPort

Syntax	INTEGER(1..2147483647)
Access	read-only
Status	mandatory
Description	The UDP port to send traps to for this host. Normally this would be the standard trap port (162). This object is an index and must be specified to create a row in this table.
Sequence	`::= { trapRegEntry 2 }`

trapRegFilter

Syntax	FcEventSeverity
Access	read-write
Status	mandatory
Description	This value defines the trap severity filter for this trap host. The connUnit will send traps to this host that have a severity level less than or equal to this value. The default value of this object is 'warning'.
Sequence	`::= { trapRegEntry 3}`

trapRegRowState

Syntax

INTEGER{

rowDestroy(1),	Remove row from table.
rowInactive(2),	Row exists, but TRAPs disabled
rowActive(3)	Row exists and is enabled for sending traps

}

Access: read-write

Status: mandatory

Description: Specifies the state of the row.

rowDestroy
READ: Can never happen.

WRITE: Remove this row from the table.

rowInactive
READ: Indicates that this row does exist, but that traps are not enabled to be sent to the target.

WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are not enabled to be sent to the target. If the row already existed, then traps are disabled from being sent to the target.

rowActive
READ: Indicates that this row exists, and that traps

are enabled to be sent to the target.

WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are enabled to be sent to the target. If the row already exists, then traps are enabled to be sent to the target.

A value of rowActive or rowInactive must be specified to create a row in the table.

Sequence `::= { trapRegEntry 4}`

Related traps

connUnitStatusChange

ENTERPRISE fcmgmt

VARIABLES { connUnitStatus, connUnitState }

Description — The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.

```
::= 1
```

connUnitPortStatusChange

ENTERPRISE fcmgmt

VARIABLES { connUnitPortStatus, connUnitPortState }

Description — The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.

Sequence

```
::= 6
```

END

A

McDATA/Nishan Common MIB

Nishan Enterprise MIB for FC Platform

Version 1.0

Revision history:

2/14/99	Initial version created.
4/25/01	Minor corrections to compile without warnings in SMICng.
11/14/02	Addition of nishanCommonAuthDevicesGroup and tables to authenticate iSCSI device logins with RADIUS servers.
02/11/03	Changed comment lines to support HP NNM. i.e replace "-" with "=".

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

Groups in the Nishan Fabric Channel PRODUCT MIB

NOTE: Common Groups are those defined without dependence upon the underlying hardware or architecture. Architecture Groups provide visibility to architecture specific items such as:

- Stackability
- Number of ports per unit
- Number of slots per unit

- Number of units per stack
- Architecture specific statistics

Chip Groups provide visibility to hardware specific items such as:

- Reading/Writing registers within the switch ASIC
- Exposing modes of the chip which are not used normally

The following groups are supported for the Nishan Enterprise MIB:

Group	Description
nishanCommonGroup	Common groups
nishanCommonVerGroup	Retrieve version information
nishanCommonIPGroup	Configure and retrieve IP connectivity items
nishanCommonAuthGroup	Configure trusted SNMP hosts, communities and RADIUS servers
nishanCommonTrapGroup	Configure SNMP traps
nishanCommonLoadGroup	Configure file download/upload parameters
nishanCommonMiscGroup	Configure and retrieve miscellaneous items
nishanArchGroup	Architecture specific groups
nishanArchIfaceGroup	Configure and retrieve interface info
nishanChipGroup	Chipset specific groups

Nishan Common MIB Definitions

```
IMPORTS
        MODULE-IDENTITY,
        Counter32,
        IpAddress - FROM SNMPv2-SMI
        DisplayString - FROM RFC1213-MIB
        nishan - FROM NISHAN-SMI
        InetAddressType,
        InetAddress - FROM INET-ADDRESS-MIB;

Last-Updated    0507270000Z
```

Organization McDATA Corporation

Contact-Info McDATA Corporation

380 Interlocken Crescent

Broomfield, CO 80021 USA

Tel : 1 720 558-8000

Fax : 1 720 558-3860

email : information@mcdata.com"

Description The private MIB for McDATA SAN Router Management.

Revision 0509190000Z

Description Released for E/OSi version 4.7.

Revision 0507270000Z

Description Improve descriptions of several variables. No functional changes.

Revision 0505050000Z

Description Released for E/OSi version 4.7.

```
::=  {  nishan  2  }
```

"MacAddress" - A 6 octet address in the "canonical" order defined by IEEE 802.1a, i.e., as if it were transmitted least significant bit first.

MacAddress ::= OCTET STRING (SIZE (6))

Textual Conventions (i.e., these conventions do not affect object encoding) "DURABLE" (Non-Standard): Objects that are saved across a system reset and/or power cycle are noted as "DURABLE" for convenience in the Description section of the object definition.

To define "DURABLE" object "factory default values", use the following rules to modify the "DURABLE line(s) in the Description section of the object definition (see examples below):

1. Curly braces(i.e., '{', '}') containing the default value(s) must follow '"DURABLE:' separated by a space or start on the next line.
2. There must be a space separating the curly brace (left or right) and the default value(s).
3. Strings are denoted using single quotes (e.g., 'public'). Do NOT use single or double quotes within a string. Strings also must NOT include curly braces.

4. Decimal INTEGER values must NOT use single quotes. Hexadecimal and binary values MUST use single quotes followed by a lower or upper case character to indicate hexadecimal (h) or binary (b) (e.g., 'FACE'H).
5. The default value may span more than one line (i.e., parser looks for the closing curly brace). Extra white space will be removed. If possible, to avoid confusion, put the default value on a single line.
6. Multiple default values can be included within the curly brace set. Multiple values must be separated using commas with no preceding or succeeding spaces; it is OK to use a 'Return' after the comma to allow the new item to start on the next line. Do NOT use a comma after the last default value.
7. To set all values use ':all' after parameter (e.g., enable:all). If ':all' is used on the last value (e.g., enable, enable, disable:all), the remaining items are set to the last supplied value.
8. The actual Description text should start on the next line after DURABLE: or, if a default value is used, the next line after the right curly brace for the default value.
9. DURABLE objects WITHOUT an explicit factory default value in curly braces, default to zero, NULL, or the lowest enumerated value. This also applies to the remaining table entries when only some of the defaults are defined.

Valid Examples:

decimal INTEGER examples:
"DURABLE: { 60 } ;single default
"DURABLE: { 1,3,7,11 } ;4 defaults
"DURABLE: { 60:all } ;all entries use 60
"DURABLE: { 1,2,30:all } ;3rd thru end use 30

hexadecimal INTEGER examples:
"DURABLE: { 'face'h }
"DURABLE: { 'face'H }
"DURABLE: { 'face'H:all }
"DURABLE: { 'deaf'h,'face'h } ;2 defaults

binary INTEGER examples:
"DURABLE: { '10101111'b }
"DURABLE: { '0010110100101101'B }

enumerated value examples:
"DURABLE: { disable }

"DURABLE: { enable,enable } ;2 defaults
"DURABLE: { enable:all } ;enable all entries
"DURABLE: { enable,disable:all } ;enable 1st; disable rest
MacAddress (OCTET STRING) example:
"DURABLE: { '0123456789ab'H }
OBJECT IDENTIFIER example:
"DURABLE: { sysDescr }
IpAddress example:
"DURABLE: { 'c0210415'h }
DisplayString examples:
"DURABLE: { 'Nishan Product' }
"DURABLE: { 'public','private' } ;2 defaults; rest null

"NONVOLATILE" (Non-Standard):
Objects that are saved across a system reset and/or power cycle and are not owned by this MIB are noted as "NONVOLATILE" for convenience in the Description section of the object definition. These objects eventually are defined "DURABLE" by some MIB in the system and are "passed through" in this MIB. The same Syntax rules apply to the NONVOLATILE as to the DURABLE. Sample Nishan FC Product Enterprise MIB Reference Switch.

NOTE: Using Wind River enterprise ID (731) for illustration purposes the customer must replace 731 with the registered enterprise identifier for his company and/or product

Nishan Enterprise MIB Extensions

nishanCommonGroup	OBJECT IDENTIFIER	::= {nishanCommon 1 }
nishanCommonVerGroup	OBJECT IDENTIFIER	::={nishanCommonGroup 1 }
nishanCommonIPGroup	OBJECT IDENTIFIER	::={nishanCommonGroup 2 }
nishanCommonAuthGroup	OBJECT IDENTIFIER	::= {nishanCommonGroup 3 }
nishanCommonTrapGroup	OBJECT IDENTIFIER	::= { nishanCommonGroup 4 }
nishanCommonLoadGroup	OBJECT IDENTIFIER	::= { nishanCommonGroup 5 }
nishanCommonMiscGroup	OBJECT IDENTIFIER	::= { nishanCommonGroup 6 }

```
nishanArchGroup          OBJECT IDENTIFIER    ::= { nishanCommon 2 }
nishanArchIfaceGroup     OBJECT IDENTIFIER    ::= { nishanArchGroup 1 }
nishanChipGroup          OBJECT IDENTIFIER    ::= { nishanCommon 3 }
```

Common Group

Nishan Extension MIB, Version Group

This group enables the retrieval of version information for various components in the system. Some version numbers have the concept of "Major.Minor" designators.

In these version numbers, the Major designator represents the primary release version, while the Minor designator represents the secondary version, incremented for bug-fixes or non-critical modifications. For example, the version number 1.2 has "1" as the Major version number and "2" as the minor version number.

nishanCommonVerMIBMajor

Syntax	INTEGER(0..2147483647)
Max-Access	read-only
Status	obsolete
Description	Get this MIB Major version number. This number should match the Major version given in the documentation header in the beginning text of this MIB. No longer supported in current E/OSi releases.
Sequence	`::= { nishanCommonVerGroup 1 }`

nishanCommonVerMIBMinor

Syntax	INTEGER(0..2147483647)
Max-Access	read-only
Status	obsolete
Description	Get this MIB Minor version number. This number should match the Minor version given in the documentation header in the beginning text of this MIB. No longer supported in current E/OSi releases.
Sequence	`::= { nishanCommonVerGroup 2 }`

nishanCommonVerBootSWMajor

Syntax	INTEGER(0..2147483647)

Max-Access	read-only
Status	current
Description	Get the Boot Software Major version number. This number references the Boot ROM module.
Sequence	`::= { nishanCommonVerGroup 3 }`

nishanCommonVerBootSWMinor

Syntax	INTEGER(0..2147483647)
Max-Access	read-only
Status	current
Description	Get the Boot Software Minor version number. This number references the Boot ROM module.
Sequence	`::= { nishanCommonVerGroup 4 }`

nishanCommonVerAppSWMajor

Syntax	INTEGER(0..2147483647)
Max-Access	read-only
Status	current
Description	Get the Application Software Major version number. This number references the core modules as a whole.
Sequence	`::= { nishanCommonVerGroup 5 }`

nishanCommonVerAppSWMinor

Syntax	INTEGER(0..2147483647)
Max-Access	read-only
Status	current
Description	Get the Application Software Minor version number. This number references the core modules as a whole.
Sequence	`::= { nishanCommonVerGroup 6 }`

Nishan Extension MIB, IP Connectivity Information

This group is meant to be a collection of controls and data that configure the device for IP connectivity.

---

nishanCommonIPMACAddr

Syntax	MacAddress
Max-Access	read-write
Status	current
Description	Get the agent MAC address.
Sequence	`::= { nishanCommonIPGroup 1 }`

---

nishanCommonIPIpAddress

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	The current inband IP Address for the SAN Router. Setting this variable has no effect on the actual value used in the SAN Router. Use nishanCommonIPIpAddressOnNextReset instead to change the SAN Router's inband address.
Sequence	`::= { nishanCommonIPGroup 2 }`

---

nishanCommonIPGateAddress

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	The current default gateway address for the SAN Router. A value of 0.0.0.0 indicates that no default gateway is set. Setting this variable has no effect on the actual value used in the SAN Router. Use nishanCommonIPGateAddressOnNextReset instead to change the SAN Router's default gateway address.
Sequence	`::= { nishanCommonIPGroup 3 }`

nishanCommonIPNetMask

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	The current subnet mask for the inband address of the SAN Router. Setting this variable has no effect on the actual value used in the SAN Router. Use nishanCommonIPNetMaskOnNextReset instead to change the SAN Router's subnet mask.
Sequence	`::= { nishanCommonIPGroup 4 }`

nishanCommonIPStatus

Syntax	INTEGER { notModified(1), modified(2), restore(3), apply(4) }
Max-Access	read-write
Status	obsolete
Description	This object is used to synchronize the modification of the IP parameters used by the protocol stack based on the nishanCommonIPGroup MIB objects. This allows the user to change nishanCommonIPMACAddr, nishanCommonIPIpAddress, nishanCommonIPGateAddress, and nishanCommonIPNetMask, and then apply the changes to the unit using apply(4). If nishanCommonIPStatus returns not_modified(1), no modifications were made to any nishanCommonIPGroup MIB objects. If nishanCommonIPStatus returns modified(2), one or more objects were changed, but have not been applied. Both not_modified(1) and modified(2) are read-only values; the agent returns a SNMP_BADVALUE for sets using these values. Both restore(3) and apply(4) are valid set values. If one or more of the nishanCommonIPGroup objects were modified, the user can restore the parameters to a mirror of the NVM values using restore(3). This variable is no longer supported in current E/OSi releases.
Sequence	`::= { nishanCommonIPGroup 5 }`

nishanCommonIPIpAddressOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	Sets the inband IP Address for the SAN Router. Changes take effect after a SAN Router reset.
Sequence	`::= { nishanCommonIPGroup 6 }`

nishanCommonIPGateAddressOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	Sets the Default Gateway Address for the SAN Router. A value of 0.0.0.0 indicates that no default gateway is set. Changes take effect after a SAN Router reset.
Sequence	`::= { nishanCommonIPGroup 7 }`

nishanCommonIPNetMaskOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	Sets the IP Subnet Mask for the SAN Router's inband IP address. Changes take effect after a SAN Router reset.
Sequence	`::= { nishanCommonIPGroup 8 }`

Nishan Extension MIB, Authentication Group

Here begins the community and host tables which provide security to the SNMPv1 agent. SNMPv1 typically has no security aspect. These tables grant that ability to the agent. This security aspect can be enabled or disabled through the nishanCommonMiscSnmpSecurity object in the nishanCommonMiscGroup.

The nishanCommonAuthHostTable lists the hosts which have access to the agent externally (nodes on the network). These hosts must belong to one of the pre-defined communities in nishanCommonAuthCommTable.

nishanCommonAuthCommTable is a list of valid communities recognized by the agent and the permissions of those communities as to READ or READ-WRITE. READ permission grants to the member of a community the ability to GET information. READ-WRITE permission grants to the member of a community the ability to GET and SET information.

EXAMPLES:
A Representative Authentication Community Table
nishanCommonAuthCommTable

Index	Name	Perm	State
1	public	2 (RW)	1 (enable) |
2	netman	2 (RW)	2 (disable) |
3	private	1 (RO)	2 (disable) |
.	.	.	.
.	.	.	.
.	.	.	.
	7		
8	admin	2 (RW)	1 (enable)

A Representative Authentication Host Table

nishanCommonAuthHostTable

Index	Name	ComonName	IP Address	IP Mask	State
1	host1	public	1.2.3.4	255.255.255.0	2 (disable)
2	host2	public	1.2.2.4	255.255.255.0	1 (enable)
3	ntbox	netman	2.3.4.5	255.255.255.0	1 (enable)
4	sunbox	xyz	3.4.5.6	255.255.255.0	(enable)
8	macbox	admin	4.3.2.1	255.255.255.0	3 (delete)

1) "host1", which belongs to community "public" attempts access to the agent. This access is disallowed because "host1" is disabled in the nishanCommonAuthHostTable.

2) "ntbox", which belongs to community "netman" attempts access to the agent. This access is disallowed because the community "netman" is disabled.

3) "sunbox", which belongs to community "xyz" attempts access to the agent. This access is disallowed because the community "xyz" does not exist.

4) "host2", which belongs to community "public" attempts SET access to the agent. This access is permitted.

---

nishanCommonAuthHostTableEntriesMax

Syntax	INTEGER (1..8)
Max-Access	read-write
Status	current
Description	NONVOLATILE: { 8 } The maximum possible number of entries in the nishanCommonAuthHostTable. This value equates to the total number of trusted Hosts which potentially may be recognized.
Sequence	`::= { nishanCommonAuthGroup 1 }`

---

nishanCommonAuthCommTableEntriesMax

Syntax	INTEGER (1..8)
Max-Access	read-write
Status	current
Description	NONVOLATILE: { 8 } The maximum possible number of entries in the nishanCommonAuthCommTable. This value equates to the total number of communities which potentially may be recognized.
Sequence	`::= { nishanCommonAuthGroup 2 }`

nishanCommonAuthCommTable

Syntax	SEQUENCE OF nishanCommonAuthCommEntry
Max-Access	not-accessible
Status	current
Description	A list of valid SNMP Community Entries.
Sequence	`::= { nishanCommonAuthGroup 3 }`

nishanCommonAuthCommEntry

Syntax	nishanCommonAuthCommEntry
Max-Access	not-accessible
Status	current
Description	A list of SNMP Community properties. INDEX { nishanCommonAuthCommIndex }

Sequence	`::= { nishanCommonAuthCommTable 1 }`
	NishanCommonAuthCommEntry ::= SEQUENCE {
	nishanCommonAuthCommIndex INTEGER,
	nishanCommonAuthCommName DisplayString,
	nishanCommonAuthCommPerm INTEGER,
	nishanCommonAuthCommState INTEGER }

nishanCommonAuthCommIndex

Syntax	INTEGER (1..8)
Max-Access	read-only
Status	current
Description	An index that uniquely identifies an ordered entry in the SNMP Community Table.
Sequence	`::= { nishanCommonAuthCommEntry 1 }`

nishanCommonAuthCommName

Syntax	DisplayString (SIZE (1..16))

Max-Access read-write

Status current

Description NONVOLATILE: { 'public','private' }

The name of this SNMP Community Entry, e.g. public.

Sequence `::= { nishanCommonAuthCommEntry 2 }`

---

nishanCommonAuthCommPerm

Syntax INTEGER {permRO(1), permRW(2)}

Max-Access read-write

Status current

Description NONVOLATILE: { perm-ro,perm-rw }

The Read/Write (GET/SET) permission for the community. A (1) indicates perm-ro (GET) permission. A (2) indicates perm-wr (GET/SET) permission.

Sequence `::= { nishanCommonAuthCommEntry 3 }`

---

nishanCommonAuthCommState

Syntax INTEGER {

enable(1),

disable(2),

delete(3),

invalid(4) read-only, set returns SNMP_BADVALUE}

Max-Access read-write

Status current

Description NONVOLATILE: { enable,enable,invalid:all }

The state of this community entry. Enable (1) defines an entry that is valid. Disable (2) defines an entry that is valid but not enabled. Delete (3) removes an entry from the table. Invalid (4) defines an entry that is no longer valid (e.g., an entry that was deleted) and should be ignored. An attempt to set a value of invalid (4) causes the agent to return SNMP_BADVALUE.

Sequence `::= { nishanCommonAuthCommEntry 4 }`

nishanCommonAuthHostTable

Syntax	SEQUENCE OF nishanCommonAuthHostEntry
Max-Access	not-accessible
Status	current
Description	A list of valid SNMP Trusted Host Entries.
Sequence	`::= { nishanCommonAuthGroup 5 }`

nishanCommonAuthHostEntry

Syntax	nishanCommonAuthHostEntry
Max-Access	not-accessible
Status	current
Description	A list of SNMP Host properties. INDEX { nishanCommonAuthHostIndex }

Sequence	`::= { nishanCommonAuthHostTable 1 }`
	NishanCommonAuthHostEntry ::= SEQUENCE {
	nishanCommonAuthHostIndex INTEGER,
	nishanCommonAuthHostName DisplayString,
	nishanCommonAuthHostCommName DisplayString,
	nishanCommonAuthHostIpAddress IpAddress,
	nishanCommonAuthHostIpMask IpAddress,
	nishanCommonAuthHostState INTEGER }

nishanCommonAuthHostIndex

Syntax	INTEGER (1..8)
Max-Access	read-only
Status	current
Description	An index that uniquely identifies an ordered entry in the SNMP Trusted Host Table.
Sequence	`::= { nishanCommonAuthHostEntry 1 }`

nishanCommonAuthHostName

Syntax	DisplayString (SIZE (1..16))
Max-Access	read-write
Status	current
Description	NONVOLATILE: The name of this SNMP Trusted Host Entry, e.g. host1.
Sequence	`::= { nishanCommonAuthHostEntry 2 }`

nishanCommonAuthHostCommName

Syntax	DisplayString (SIZE (1..16))
Max-Access	read-write
Status	current
Description	NONVOLATILE: The name of this SNMP Trusted Host Entry's community, e.g., public.
Sequence	`::= { nishanCommonAuthHostEntry 3 }`

nishanCommonAuthHostIpAddress

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	NONVOLATILE: The IP Address for the host. This identified host has permission to access information in this device's MIB tables, providing the SNMP security toggle is on (see nishanCommonMiscSNMPSecurity).
Sequence	`::= { nishanCommonAuthHostEntry 4 }`

nishanCommonAuthHostIpMask

Syntax	IpAddress
Max-Access	read-write

Status current

Description NONVOLATILE: { 'ffffff00'H }

The Network Mask for the trusted Host entry.

Sequence `::= { nishanCommonAuthHostEntry 5 }`

---

nishanCommonAuthHostState

Syntax INTEGER {enable(1),
disable(2),
delete(3),
invalid(4) -- read-only, set returns SNMP_BADVALUE
}

Max-Access read-write

Status current

Description NONVOLATILE: { enable,invalid:all }

The state of this trusted host entry. Enable (1) defines an entry that is valid. Disable (2) defines an entry that is valid but not enabled. Delete (3) removes an entry from the table. Invalid (4) defines an entry that is no longer valid (e.g., an entry that was deleted) and should be ignored. An attempt to set a value of invalid (4) causes the agent to return SNMP_BADVALUE.

Sequence `::= { nishanCommonAuthHostEntry 6 }`

Nishan Extension MIB, Authenticating device logins

This group configures parameters used to authenticate devices, users, applications. A Representative RADIUS Authentication Table.

nishanCommonAuthRadiusServerTable

Index	Type	Address Type	Address	Port	Credential	Timeout	Retries
1	Primary	ipv4	1.2.3.4	1812	nishanLogin	1s	2
2	Primary	ipv4	4.2.3.1	23412	nishanLogin	1s	2

This table allows the ability to add more parameters per server if required in the future.

nishanCommonAuthDevicesGroup OBJECT IDENTIFIER ::= { nishanCommonAuthGroup 6 }

nishanCommonAuthDevicesMaxRadiusServers

Syntax	INTEGER (1..8)
Max-Access	read-only
Status	current
Description	The maximum possible number of entries in the nishanCommonAuthRadiusServerTable. This value equates to the total number of potential RADIUS servers that the switch could authenticate the iSCSI logins against. DEFVAL { 2 }
Sequence	`::= { nishanCommonAuthDevicesGroup 1 }`

nishanCommonAuthRadiusServerTable

Syntax	SEQUENCE OF nishanCommonAuthRadiusServerEntry
Max-Access	not-accessible
Status	current
Description	This table lists the entries for the RADIUS servers, and is used by the NISHAN TCP ports, which also act as a RADIUS clients, to authenticate iSCSI logins from initiators. The user can only modify the contents of the RADIUS server entries and cannot create/delete server-rows. For this release, the table has a primary and a secondary RADIUS server, with the ip address initialized to 0.0.0.0.
Sequence	`::= { nishanCommonAuthDevicesGroup 2 }`

nishanCommonAuthRadiusServerEntry

Syntax	nishanCommonAuthRadiusServerEntry
Max-Access	not-accessible
Status	current

Description A list of RADIUS Servers and their properties. To indicate that an entry is not in use, set the nishanCommonAuthRadiusServerIPAddress to 0.0.0.0.

INDEX { nishanCommonAuthRadiusServerIndex }

Sequence `::= { nishanCommonAuthRadiusServerTable 1 }`

NishanCommonAuthRadiusServerEntry ::= SEQUENCE {

nishanCommonAuthRadiusServerIndex	INTEGER,
nishanCommonAuthRadiusServerType	INTEGER,
nishanCommonAuthRadiusServerAddressType	InetAddressType,
nishanCommonAuthRadiusServerIPAddress	InetAddress,
nishanCommonAuthRadiusServerUdpPort	INTEGER,
nishanCommonAuthRadiusServerCredential	OCTET STRING,
nishanCommonAuthRadiusServerTimeout	INTEGER,
nishanCommonAuthRadiusServerRetries	INTEGER }

---

nishanCommonAuthRadiusServerIndex

Syntax INTEGER (1..8)

Max-Access read-only

Status current

Description An index that uniquely identifies an ordered entry in the RADIUS servers table. The RADIUS client on the NISHAN switch will contact all the primary RADIUS servers first sequentially and then contact the secondary servers if the primary response fails.

Sequence `::= { nishanCommonAuthRadiusServerEntry 1 }`

---

nishanCommonAuthRadiusServerType

Syntax INTEGER { primary(1), secondary(2) }

Max-Access read-write

Status current

Description DURABLE: {1:all}

The type of this RADIUS server, primary or secondary. The primary RADIUS server is contacted first and if no response is received within

a timeout period, then the secondary server is contacted. Atleast one of the servers should be set to primary.

DEFVAL { primary }

Sequence `::= { nishanCommonAuthRadiusServerEntry 2 }`

---

nishanCommonAuthRadiusServerAddressType

Syntax InetAddressType

Max-Access read-write

Status current

Description DURABLE: {1:ipv4}

The Inet address type of the RADIUS server. For now, only IPv4 is supported.

DEFVAL { ipv4 }

Sequence `::= { nishanCommonAuthRadiusServerEntry 3 }`

---

nishanCommonAuthRadiusServerIPAddress

Syntax InetAddress

Max-Access read-write

Status current

Description DURABLE:

The IP address of the RADIUS server to which this switch will send device-login authentication requests. To indicate that this server is not in use, set the value to 0.0.0.0

Sequence `::= { nishanCommonAuthRadiusServerEntry 4 }`

---

nishanCommonAuthRadiusServerUdpPort

Syntax INTEGER (0..65535)

Max-Access read-write

Status current

Description DURABLE:

The UDP port on the RADIUS server to send the authentication requests. RFC 2865 defaults this port to 1812, but could be different for other implementations.

DEFVAL { 1812 }

Sequence `::= { nishanCommonAuthRadiusServerEntry 5 }`

---

nishanCommonAuthRadiusServerCredential

Syntax OCTET STRING (SIZE (255))

Max-Access read-write

Status current

Description DURABLE:

The secret code defined by RFC 2865 to use to communicate with the RADIUS server. Nishan switches only support CHAP authentication in SW release 3.1. This variable is not required for CHAP authentication.

Sequence `::= { nishanCommonAuthRadiusServerEntry 6 }`

---

nishanCommonAuthRadiusServerTimeout

Syntax INTEGER (1..65535)

Max-Access read-write

Status current

Description DURABLE: {1}

The timeout value in seconds after which the next RADIUS server in the table is contacted for login authentication.

DEFVAL { 1 }

Sequence `::= { nishanCommonAuthRadiusServerEntry 7 }`

---

nishanCommonAuthRadiusServerRetries

Syntax INTEGER (1..5)

Max-Access read-write

Status current

Description DURABLE: {1}

The number of retries after which the next RADIUS server in the table is contacted for login authentication.

DEFVAL { 1 }

Sequence `::= { nishanCommonAuthRadiusServerEntry 8 }`

Nishan Extension MIB, Trap Group

Here begins the trap and destination tables which provide configured SNMP Trap capability to the SNMPv1 agent. A "TestTrap" facility is provided to test the trap subsystem. See "nishanCommonTrapTest".

The nishanCommonTrapCommTable lists the communities which hosts in the nishanCommonTrapDestTable belong to. The nishanCommonTrapCommTable defines types of SNMP traps which destinations belonging to these communities are sent.

EXAMPLES:

A Representative Trap Community Table

nishanCommonTrapCommTable

Index	Name	Cold Start	Link Down	Link Up	Auth	1493	RMON	Spec	State
1	public	1	1	1	2	1	1	1	1
2	netman	2	1	1	1	1	1	1	1
3	private	2	1	1	1	1	1	1	1
8	admin	2	1	1	1	1	1	1	1

A Representative Trap Host (Destination) Table

nishanCommonAuthHostTable

Index	Name	CommName	IPAddress	IPMask	State
1	host1	public	1.2.3.4	255.255.255.0	2 (disable)
2	host2	public	1.2.2.4	255.255.255.0	(enable) |
3	ntbox	netman	2.3.4.5	255.255.255.0	1 (enable) |

4	sunbox	xyz	3.4.5.6	255.255.255.0	(enable) |
8	macbox	admin	4.3.2.1	255.255.255.0	3 (delete)

nishanCommonTrapTest

Syntax	INTEGER {noop (1),coldstart (2), linkdown (3), linkup (4), authentication (5),oemspecific (6)}
Max-Access	read-write
Status	current
Description	This object sends a particular kind of SNMP Trap, as selected by type, to the SNMP manager to verify proper operation of the Trap Subsystem. A GET always returns the noop (1) value.
Sequence	`::= { nishanCommonTrapGroup 1 }`

nishanCommonTrapCommTableEntriesMax

Syntax	INTEGER (1..8)
Max-Access	read-write
Status	current
Description	NONVOLATILE: { 8 } The total number of entries in the nishanCommonTrapCommTable. This value equates to the total number of communities which potentially can be recognized.
Sequence	`::= { nishanCommonTrapGroup 2 }`

nishanCommonTrapDestTableEntriesMax

Syntax	INTEGER (1..8)
Max-Access	read-write
Status	current
Description	NONVOLATILE: { 8 }

The total number of entries in the nishanCommonTrapDestTable. This equates to the total number of hosts which can potentially be sent SNMP traps.

Sequence `::= { nishanCommonTrapGroup 3 }`

nishanCommonTrapCommTable

Syntax SEQUENCE OF NishanCommonTrapCommEntry

Max-Access not-accessible

Status current

Description A list of valid SNMP Community Entries used for Traps.

Sequence `::= { nishanCommonTrapGroup 4 }`

nishanCommonTrapCommEntry

Syntax NishanCommonTrapCommEntry

Max-Access not-accessible

Status current

Description A list of SNMP Trap Community properties.

INDEX { nishanCommonTrapCommIndex }

Sequence `::= { nishanCommonTrapCommTable 1 }`

NOTE: WarmStart and EgpNeighborLoss Traps are not supported.

NishanCommonTrapCommEntry ::= SEQUENCE {

nishanCommonTrapCommIndex	INTEGER,
nishanCommonTrapCommName	DisplayString,
nishanCommonTrapCommColdStart	INTEGER,
nishanCommonTrapCommLinkDown	INTEGER,
nishanCommonTrapCommLinkUp	INTEGER,
nishanCommonTrapCommAuthentication	INTEGER,
nishanCommonTrapCommBridge	INTEGER,

nishanCommonTrapCommRMON	INTEGER,
nishanCommonTrapCommOEMSpecific	INTEGER,
nishanCommonTrapCommState	INTEGER}

nishanCommonTrapCommIndex

Syntax	INTEGER (1..8)
Max-Access	read-only
Status	current
Description	An index that uniquely identifies an ordered entry in the SNMP Community Trap Table.
Sequence	`::= { nishanCommonTrapCommEntry 1 }`

nishanCommonTrapCommName

Syntax	DisplayString (SIZE (1..16))
Max-Access	read-write
Status	current
Description	NONVOLATILE: { 'public' } The name of this SNMP Trap Community Entry, e.g. public.
Sequence	`::= { nishanCommonTrapCommEntry 2 }`

nishanCommonTrapCommColdStart

Syntax	INTEGER {enable(1),disable(2)}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP ColdStart Trap for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 3 }`

nishanCommonTrapCommLinkDown

Syntax	INTEGER {enable(1),disable(2)}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP Link Down traps for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 4 }`

nishanCommonTrapCommLinkUp

Syntax	INTEGER {enable(1), disable(2)}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP Link Up traps for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 5 }`

nishanCommonTrapCommAuthentication

Syntax	INTEGER {enable(1),disable(2)}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP Authenticaiton traps for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 6 }`

nishanCommonTrapCommBridge

Syntax	INTEGER {enable(1),disable(2)}
Max-Access	read-write

Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP Bridge MIB traps for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 7 }`

nishanCommonTrapCommRMON

Syntax	INTEGER {enable(1),disable(2)}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP RMON traps for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 8 }`

nishanCommonTrapCommOEMSpecific

Syntax	INTEGER {enable(1),disable(2)}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all } Enable (1) or Disable (2) the SNMP Enterprise traps for this SNMP community.
Sequence	`::= { nishanCommonTrapCommEntry 9 }`

nishanCommonTrapCommState

Syntax	INTEGER {enable(1), disable(2), delete(3), invalid(4) -- read-only, set returns SNMP_BADVALUE}
Max-Access	read-write
Status	current
Description	NONVOLATILE: { enable:all }

The state of this community entry. Enable (1) defines an entry that is valid. Disable (2) defines an entry that is valid but not enabled. Delete (3) removes an entry from the table. Invalid (4) defines an entry that is no longer valid (e.g., an entry that was deleted) and should be ignored. An attempt to set a value of invalid (4) causes the agent to return SNMP_BADVALUE.

Sequence `::= { nishanCommonTrapCommEntry 10 }`

---

nishanCommonTrapDestTable

Syntax SEQUENCE OF NishanCommonTrapDestEntry

Max-Access not-accessible

Status current

Description A list of valid SNMP Trap Trusted Host Destination entries.

Sequence `::= { nishanCommonTrapGroup 5 }`

---

nishanCommonTrapDestEntry

Syntax NishanCommonTrapDestEntry

Max-Access not-accessible

Status current

Description A list of SNMP Trap Trusted Host Destination properties.

INDEX { nishanCommonTrapDestIndex }

Sequence `::= { nishanCommonTrapDestTable 1 }`

NishanCommonTrapDestEntry ::= SEQUENCE {

nishanCommonTrapDestIndex	INTEGER,
nishanCommonTrapDestName	DisplayString,
nishanCommonTrapDestCommName	DisplayString,
nishanCommonTrapDestIpAddress	IpAddress,
nishanCommonTrapDestIpMask	IpAddress,
nishanCommonTrapDestState	INTEGER}

---

nishanCommonTrapDestIndex

Syntax INTEGER (1..8)

Max-Access	read-only
Status	current
Description	An index that uniquely identifies an ordered entry in the SNMP Trusted Host Destination Table.
Sequence	`::= { nishanCommonTrapDestEntry 1 }`

nishanCommonTrapDestName

Syntax	DisplayString (SIZE (1..16))
Max-Access	read-write
Status	current
Description	NONVOLATILE: The name of this SNMP Trusted Host Destination Entry, e.g. Admin.
Sequence	`::= { nishanCommonTrapDestEntry 2 }`

nishanCommonTrapDestCommName

Syntax	DisplayString (SIZE (1..16))
Max-Access	read-write
Status	current
Description	NONVOLATILE: The name of this SNMP Trusted Host Destination Entry's community, e.g., public, or trap.
Sequence	`::= { nishanCommonTrapDestEntry 3 }`

nishanCommonTrapDestIpAddress

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	NONVOLATILE: The IP Address for the host. This identified host has permission to access information in this device's MIB tables, providing the SNMP security toggle is on (see nishanCommonMiscSNMPSecurity).

Sequence `::= { nishanCommonTrapDestEntry 4 }`

nishanCommonTrapDestIpMask

Syntax IpAddress

Max-Access read-write

Status current

Description NONVOLATILE: { 'ffffff00'H:all }

The Network Mask for the trusted Host Destination entry.

Sequence `::= { nishanCommonTrapDestEntry 5 }`

nishanCommonTrapDestState

Syntax INTEGER {

enable(1),

disable(2),

delete(3),

invalid(4) -- read-only, set returns SNMP_BADVALUE

}

Max-Access read-write

Status current

Description NONVOLATILE: { enable,disable:all }

The state of this trusted trap destination entry. Enable (1) defines an entry that is valid. Disable (2) defines an entry that is valid but not enabled. Delete (3) removes an entry from the table. Invalid (4) defines an entry that is no longer valid (e.g., an entry that was deleted) and should be ignored. An attempt to set a value of invalid (4) causes the agent to return SNMP_BADVALUE.

Sequence `::= { nishanCommonTrapDestEntry 6 }`

Begin 3-22

Nishan Extension MIB, Load TFTP Group

This group is meant to be a collection of controls and data that determine the file download and upload configuration and mode.

nishanCommonLoadTftpAddress

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	Set IP Address for the TFTP server used for downloading and uploading files.
Sequence	`::= { nishanCommonLoadGroup 1 }`

nishanCommonLoadTftpFileName

Syntax	DisplayString (SIZE (0..64))
Max-Access	read-write
Status	current
Description	Set TFTP file path and name.
Sequence	`::= { nishanCommonLoadGroup 2 }`

nishanCommonLoadType

Syntax	INTEGER {application(1),boot(2),configuration(3)}
Max-Access	read-write
Status	current
Description	The type of file to download or upload upon an nishanCommonLoadExecute. ***WARNING: The boot should only be downloaded when absolutely required (e.g., if power is removed during the boot Flash write operation, the agent cannot be recovered).***
Sequence	`::= { nishanCommonLoadGroup 3 }`

nishanCommonLoadExecute

Syntax INTEGER {

noop(1), -- no operation

download(2), -- Application, Boot, or Configuration

upload(3) -- Configuration only

}

Max-Access read-write

Status current

Description
Execute file download or upload procedure. A SET of this object starts the load procedure. Note that the Application and Boot images can only be downloaded. The Configuration file can be uploaded and downloaded. A GET of this object will return a noop (1). Use nishanCommonLoadExecuteStatus to determine the status of the executed load.

Sequence `::= { nishanCommonLoadGroup 4 }`

nishanCommonLoadExecuteStatus

Syntax INTEGER {

notStarted(1),

inProgress(2),

success(3),

errorConnection(4),

errorFilename(5),

errorFault(6)

Max-Access read-only

Status current

Description — Provides status on the execute file load procedure. The status return is given by one of the enumerated codes:

not-started	(download has not yet started)
in-progress	(download has not yet finished)
success	(download has finished successfully)
error-connection	(download cannot connect)
error-filename	(download has bad image path/filename)
error-fault	(download has fault condition on device).

Sequence — `::= { nishanCommonLoadGroup 5 }`

Nishan Extension MIB, Miscellaneous Information

This group is meant to be a collection of controls and data that do not require a group of their own, or fall into the category of "miscellaneous". The user may easily add simple objects in this group without destroying the integrity or definition of the group.

---

nishanCommonMiscSaveToNvm

Syntax — INTEGER {

noop(1),	-- no operation
save(2)	-- save all configuration parameters to NVM

}

Max-Access — read-write

Status — current

Description — When set to save (2), all nonvolatile configuration parameters are saved to NVM (Nonvolatile Memory). NVM save operations may be slow (e.g., erasing/writing Flash device). Consequently, all (or many) updates should be made before saving to NVM. A get operation always returns noop (1).

Sequence — `::= { nishanCommonMiscGroup 1 }`

---

nishanCommonMiscSnmpSecurityOnOff

Syntax — INTEGER {enable(1),disable(2)}

Max-Access — read-write

Status current

Description NONVOLATILE: { disable }

Enable/Disable SNMP Security. A (1) enables, a (2) disables. This object controls the application of the Community and Host tables. WARNING: If SNMP security is disabled, ANY HOST has access to the SNMP agent. See nishanCommonAuthenGroup and nishanCommonTrapGroup.

Sequence `::= { nishanCommonMiscGroup 2 }`

---

nishanCommonMiscSpanOnOff

Syntax INTEGER {enable(1),disable(2)}

Max-Access read-write

Status current

Description Enable/Disable Spanning Tree. A (1) enables, a (2) disables. When disabled, the ports of the device are placed in the forwarding mode, regardless of current Spanning Tree state. When re-enabled, the normal state transitions take place.

Sequence `::= { nishanCommonMiscGroup 3 }`

---

nishanCommonMiscBOOTPOnOff

Syntax INTEGER {enable(1),disable(2)}

Max-Access read-write

Status current

Description Enable/Disable BOOTP operation. A (1) enables, a (2) disables. When disabled, no BOOTPs are transmitted by the Application. When enabled, the Application sends up to 3 BOOTP requests at 5 second intervals.

Sequence `::= { nishanCommonMiscGroup 4 }`

---

nishanCommonMiscDHCPOnOff

Syntax INTEGER {enable(1),disable(2)}

Max-Access read-write

Status current

Description — Enable/Disable DHCP operation. A (1) enables, a (2) disables. When disabled, no DHCPs are transmitted by the Application. When enabled, the Application sends up to 3 DHCP requests at 5 second intervals.

Sequence — `::= { nishanCommonMiscGroup 5 }`

---

nishanCommonMiscBaud

Syntax — INTEGER {baud2400(1),baud9600(2),baud19200(3),baud38400(4)}

Max-Access — read-write

Status — current

Description — The Serial port BAUD Rate. Attributes are 8 Start Bits, no parity and 1 stop bits (8N1) and Hardware Flow Control. Valid values are 2400, 9600, 19200, and 38400.

Sequence — `::= { nishanCommonMiscGroup 6 }`

---

nishanCommonMiscPassword

Syntax — DisplayString (SIZE (1..16))

Max-Access — read-write

Status — current

Description — Get and Set the system password.

Sequence — `::= { nishanCommonMiscGroup 7 }`

---

nishanCommonMiscProductName

Syntax — DisplayString (SIZE (1..32))

Max-Access — read-write

Status — current

Description — Get and Set the product name of this agent's device. This name is shown in user interface screens.

Sequence — `::= { nishanCommonMiscGroup 8 }`

---

nishanCommonMiscReset

Syntax — INTEGER {

noop(1), -- no operation

reset(2), -- reset unit

resetToDefaults(3) -- reset to factory defaults and then reset unit

}

Max-Access read-write

Status current

Description Physically performs a hardware reset of the unit. Use with care. A reset (2) resets the unit, a reset-to-defaults (3) resets the nonvolatile configuration to factory defaults and then resets the unit, and noop (1) does nothing. A value of noop (1) is always returned for a GET operation.

Sequence `::= { nishanCommonMiscGroup 9 }`

Architecture Group

Interface Information

This group provides interface control and information not easily available from other MIBs. Generic switches have <unit>.<slot>.<port> topology Any port designator may be described in this manner.

The port topology supported by the Nishan FC reference platform is of the form <unit>.<port> since there are no slots. The units may be stacked up to 4 high. Each unit has 24 10/100 Ports and 2 1000 Ports. Port designators will then have the form <unit>.<port>, where unit may range from 1 through 4 and port may range from 1 through 26.

The maximum configuration is shown below:

Unit	Ports 10/100	Ports 1000	
4	< 1 - 24	25,26 >	
3	< 1 - 24	25,26 >	4 units x 26 ports
2	< 1 - 24	25,26 >	= 104 unique interfaces
1	< 1 - 24	25,26 >	

The <unit>.<port> numbering scheme translates into a logical port numbering scheme which can best be described as linear with holes. These "holes" accomodate missing units or virtual ports such as trunks. Logical ports track the physical ports on a one-to-one correspondence, with the holes being "extra" ports to be defined. For example, logical port 26 is physical unit 1, port 26. Logical port 66 is physical unit 3, port 2. The translation is shown below:

Unit	10/100	1000	Logical Ports
4	< 1 - 24	25,26 >	< 97 - 122 >
3	< 1 - 24	25,26 >	< 65 - 90 >
2	< 1 - 24	25,26 >	< 33 - 58 >
1	< 1 - 24	25,26 >	< 1 - 26 >

nishanArchIfaceTable

Syntax	SEQUENCE OF NishanArchIfaceEntry
Max-Access	not-accessible
Status	current
Description	A list of Interfaces and their properties
Sequence	`::= { nishanArchIfaceGroup 1 }`

nishanArchIfaceEntry

Syntax	NishanArchIfaceEntry
Max-Access	not-accessible
Status	current
Description	An index of interfaces. INDEX { nishanArchIfaceUnit, nishanArchIfacePort }

Sequence	`::= { nishanArchIfaceTable 1 }`
	NishanArchIfaceEntry ::= SEQUENCE {
	nishanArchIfaceUnit INTEGER,
	nishanArchIfacePort INTEGER,
	nishanArchIfaceLogicalPort INTEGER,
	nishanArchIfaceName DisplayString,

nishanArchIfaceType	INTEGER,
nishanArchIfaceEnable	INTEGER,
nishanArchIfaceSTPEnable	INTEGER,
nishanArchIfaceLink	INTEGER,
nishanArchIfaceDuplexSpeedSet	INTEGER,
nishanArchIfaceDuplexSpeedGet	INTEGER,
nishanArchIfaceTXOctetsNoErr	Counter32,
nishanArchIfaceTXPacketsNoErr	Counter32,
nishanArchIfaceRXOctetsNoErr	Counter32,
nishanArchIfaceRXPacketsNoErr	Counter32}

nishanArchIfaceUnit

Syntax	INTEGER (1..4)
Max-Access	read-only
Status	current
Description	An index that uniquely identifies a unit in the Nishan/FC Interface Table.
Sequence	`::= { nishanArchIfaceEntry 1 }`

nishanArchIfacePort

Syntax	INTEGER (1..32)
Max-Access	read-only
Status	current
Description	An index that uniquely identifies a port within the unit in the Nishan/FC Interface Table.
Sequence	`::= { nishanArchIfaceEntry 2 }`

nishanArchIfaceLogicalPort

Syntax	INTEGER (1..128)
Max-Access	read-only
Status	current

Description An index that uniquely identifies a port in the Nishan/FC Interface Table. This is a linear port number that may have holes (e.g., missing units). For this MIB, 32 port numbers are reserved for each unit in the stack. For example, port 101 is unit 4, port 5.

Sequence `::= { nishanArchIfaceEntry 3 }`

---

nishanArchIfaceName

Syntax DisplayString (SIZE (1..16))

Max-Access read-write

Status current

Description The textual name of this interface, e.g., 'John'.

Sequence `::= { nishanArchIfaceEntry 4 }`

---

nishanArchIfaceType

Syntax INTEGER {

other(1), -- none of the following

regular1822(2),

hdh1822(3),

ddnX25(4),

rfc877X25(5),

ethernetCSMACD(6),

iso88023CSMACD(7),

iso88024TokenBus(8),

iso88025TokenRing(9),

iso88026Man(10),

starLan(11),

proteon10Mbit(12),

proteon80Mbit(13),

hyperchannel(14),

fddi(15),

lapb(16),
sdlc(17),
ds1(18), -- T-1
e1(19), -- european equivalent. of T-1
basicISDN(20),
primaryISDN(21), -- proprietary serial
propPointToPointSerial(22),
ppp(23),
softwareLoopback(24),
eon(25), -- CLNP over IP [11]
ethernet3Mbit(26),
nsip(27), -- XNS over IP
slip(28), -- generic SLIP
ultra(29), -- ULTRA technologies
ds3(30), -- T-3
sip(31), -- SMDS
frameRelay(32)
}

Max-Access read-only

Status current

Description The type of interface, repeated here from the definitions in RFC1213 as a matter of convenience.

Sequence `::= { nishanArchIfaceEntry 5 }`

nishanArchIfaceEnable

Syntax INTEGER {enable(1),disable(2)}

Max-Access read-write

Status current

Description Enable (1) and Disable (2) control for the interface. For this product, this is the ONLY way to enable or disable the interface.

'ifAdminStatus' in RFC1213 and 'dot1dStpPortEnable' in RFC1493 are both implemented as 'read-only'.

Sequence `::= { nishanArchIfaceEntry 6 }`

---

nishanArchIfaceSTPEnable

Syntax INTEGER {enable(1),disable(2)}

Max-Access read-write

Status current

Description Enable (1) and Disable (2) Spanning Tree operation for this interface.

Sequence `::= { nishanArchIfaceEntry 7 }`

---

nishanArchIfaceLink

Syntax INTEGER {up(1), down(2)}

Max-Access read-only

Status current

Description The state of Link Detect on the interface.

Sequence `::= { nishanArchIfaceEntry 8 }`

---

nishanArchIfaceDuplexSpeedSet

Syntax INTEGER { autonegotiate(1), half10(2), full10(3), half100(4), full100(5), full1000(6), illegal(99) }

Max-Access read-write

Status current

Description The desired speed and duplex for the interface. If the selected control is not possible on the interface, a value of (99) illegal is returned. If the port type does NOT support the default of autonegotiate (1), then the application will initialize the port to a valid value (e.g., 1000full (6)). Note that not all controls are possible for all interfaces. For example, only full-1000 (6) is available for Gigabit Ethernet interfaces.

Sequence `::= { nishanArchIfaceEntry 9 }`

nishanArchIfaceDuplexSpeedGet

Syntax	INTEGER {unknown(1), half10(2), full10(3), half100(4), full100(5), full1000(6), illegal(99) }
Max-Access	read-only
Status	current
Description	The actual speed and duplex for the interface. If the interface is not configured to one of the acceptable values, a value of illegal (99) is returned.
Sequence	`::= { nishanArchIfaceEntry 10 }`

nishanArchIfaceTXOctetsNoErr

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of octets transmitted from the interface. This number does not include octets in error.
Sequence	`::= { nishanArchIfaceEntry 11 }`

nishanArchIfaceTXPacketsNoErr

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of packets transmitted from the interface. This number does not include packets in error.
Sequence	`::= { nishanArchIfaceEntry 12 }`

nishanArchIfaceRXOctetsNoErr

Syntax	Counter32
Max-Access	read-only
Status	current

Description	The number of octets received on the interface. This number does not include octets in error.
Sequence	`::= { nishanArchIfaceEntry 13 }`

nishanArchIfaceRXPacketsNoErr

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of packets received on the interface. This number does not include packets in error.
Sequence	`::= { nishanArchIfaceEntry 14 }`

Nishan Extension MIB: Chip Group

nishanChipStub

Syntax	INTEGER { noop(1), chipValue2(2), chipValue3(3) }
Max-Access	read-write
Status	current
Description	This object is a placeholder for the nishanChipGroup.
Sequence	`::= { nishanChipGroup 1 }`

Sample Enterprise TRAP :nishanTrap

Enterprise	nishan
Description	This trap is used for the sample enterprise trap in the nishanCommonTrapTest object and as an example for developers of custom enterprise MIBs.
Sequence	`::= 1` END

B

McDATA SAN Routing Management MIB

NISHAN-FCMGMT.MIB

McDATA SAN Routing Management MIB

This MIB contains management objects for McDATA SAN Routers. This MIB configures and monitors SAN Routing functions, in which the SAN Router imports and exports devices to multiple connected FC fabrics.

Variable descriptions use the term "R_Port" for SAN Routing ports, while the variable names use the term "eport". R_Ports are an extension of the standard E_Ports to add routing capabilities.

Revision history:

7/03/01 Initial version

7/10/01 Changes made with initial review from MIB team.

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

```
NISHAN-FCMGMT DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, IpAddress, OBJECT-TYPE

        FROM SNMPv2-SMI

DisplayString, RowStatus
```

FROM SNMPv2-TC

nishan

FROM NISHAN-SMI

WWNtype, FCIDtype

FROM NISHAN-MGT;

nishanFcMgmt MODULE-IDENTITY

Last Updated	0509190000Z
Organization	McDATA Corporation
Contact info	McDATA Corporation 380 Interlocken Crescent Broomfield, CO 80021 USA Tel : 1 720 558-8000 Fax : 1 720 558-3860 email : information@mcdata.com
Description	The private McDATA MIB for SAN Routing.
Revision	0509190000Z
Description	Released for E/OSi version 4.7."
Revision	0507140000Z
Description	Multiple description updates from Product Test & Integration. Multiple variables marked obsolete or not supported in the current release.
Revision	0107030000Z
Description	Initial public release of this MIB module.
Sequence	`::= { nishan 11 }`

R_Port Support Branch

The following is a collection of object which present information about a particular R_Port.

nishanEport OBJECT IDENTIFIER ::= { nishanFcMgmt 1 }

R_Port device list, and switch list information

The following tables shows the relationship of what switches and devices are connected to a particular R_Port.

nishanEportDeviceInfo OBJECT IDENTIFIER ::= { nishanEport 1 }

eportSwitchTable

Syntax	SEQUENCE OF eportSwitchEntry
Max-Access	not-accessible
Status	current
Description	This table contains information on the different FC switches connected to a local R_Port.
Sequence	`::= { nishanEportDeviceInfo 1 }`

eportSwitchEntry

Syntax	eportSwitchEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular switch off of a local R_Port. INDEX { eportSwitchLocalSoIPPortName, eportSwitchFCSwitchName }
Sequence	`::= { eportSwitchTable 1 }`

EportSwitchEntry ::= SEQUENCE {

eportSwitchLocalSoIPPortName	WWNtype,
eportSwitchFCSwitchName	WWNtype,
eportSwitchTrueDomainId	INTEGER,
eportSwitchSoIPPrincipalFabricPort	WWNtype,
eportSwitchFCSwitchProductName	DisplayString,
eportSwitchFCSwitchProductVersion	DisplayString,
eportSwitchIpAddress	IpAddress}

eportSwitchLocalSoIPPortName

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The logical port off of which the current FC Switch resides. The same switch can reside on more than one port.
Sequence	`::= { eportSwitchEntry 1 }`

eportSwitchFCSwitchName

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The FC Switch's WWN which resides off of a local R_Port.
Sequence	`::= { eportSwitchEntry 2 }`

eportSwitchTrueDomainId

Syntax	INTEGER (1..240)
Max-Access	read-only
Status	current
Description	The FC Switch's true domain ID. The domain ID it uses in a given SAN Island. If the operating mode is Open Fabric, the configured value is equal to (eportSwitchTrueDomainId value - 96).
Sequence	`::= { eportSwitchEntry 3 }`

eportSwitchSoIPPrincipalFabricPort

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The Principal E_Port which the following switch is eventually connected to.

Sequence ::= { eportSwitchEntry 4 }

eportSwitchFCSwitchProductName

Syntax DisplayString

Max-Access read-only

Status deprecated

Description The FC Switch's Product Name.

Sequence ::= { eportSwitchEntry 5 }

eportSwitchFCSwitchProductVersion

Syntax DisplayString

Max-Access read-only

Status deprecated

Description The FC Switch's Product Version.

Sequence ::= { eportSwitchEntry 6 }

eportSwitchIpAddress

Syntax IpAddress

Max-Access read-only

Status current

Description The IP address of the FC Switch.

Sequence ::= { eportSwitchEntry 7 }

eportDeviceTable

Syntax SEQUENCE OF eportDeviceEntry

Max-Access not-accessible

Status current

Description This table contains information on how each device on a local R_Port is connected, and which Fibre Channel Switch it is connected too.

Sequence ::= { nishanEportDeviceInfo 2 }

eportDeviceEntry

Syntax	EportDeviceEntry
Max-Access	not-accessible
Status	current
Description	Entry containing info for a particular device off of a local R_Port.
	INDEX { eportDeviceSoIPPortName, eportDeviceWwn }
Sequence	`::= { eportDeviceTable 1 }`
	EportDeviceEntry ::= SEQUENCE {
	eportDeviceSoIPPortName WWNtype,
	eportDeviceWwn WWNtype,
	eportDeviceFcSwitchWWN WWNtype,
	eportDeviceRealFCID FCIDtype,
	eportDeviceSoIPProxyFCID FCIDtype,
	eportDeviceSoIPPrincipalFabricPort WWNtype,
	eportDeviceFcSwitchFabricPort WWNtype,
	eportDeviceFcSwitchPort INTEGER}

eportDeviceSoIPPortName

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The Port Name that identifies which R_Port the device resides off of.
Sequence	`::= { eportDeviceEntry 1 }`

eportDeviceWwn

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The WWN that identifies a device that is off of a local R_Port.

Sequence `::= { eportDeviceEntry 2 }`

eportDeviceFcSwitchWWN

Syntax WWNtype

Max-Access read-only

Status current

Description The Fabric Switch Name off of which the current device resides.

Sequence `::= { eportDeviceEntry 3 }`

eportDeviceRealFCID

Syntax FCIDtype

Max-Access read-only

Status current

Description The FCID used by this device in the local SAN Island. This is the true FCID of the device.

Sequence `::= { eportDeviceEntry 4 }`

eportDeviceSoIPProxyFCID

Syntax FCIDtype

Max-Access read-only

Status current

Description The FCID used by this device in the SAN Router fabric. This is the proxy representation of the FCID on the SAN Router fabric.

Sequence `::= { eportDeviceEntry 5 }`

eportDeviceSoIPPrincipalFabricPort

Syntax WWNtype

Max-Access read-only

Status current

Description The SAN Router Fabric Port Name that represents the Principal R_Port that eventually this device connects to.

Sequence ::= { eportDeviceEntry 6 }

---

eportDeviceFcSwitchFabricPort

Syntax WWNtype

Max-Access read-only

Status obsolete

Description The FC Fabric Port Name of the FC port to which this device is connected to.

Sequence ::= { eportDeviceEntry 7 }

---

eportDeviceFcSwitchPort

Syntax INTEGER (1..65535)

Max-Access read-only

Status obsolete

Description The FC Fabric Port Number of the FC port to which this device is connected to.

Sequence ::= { eportDeviceEntry 8 }

---

eportSwitchFabricTable

Syntax SEQUENCE OF EportSwitchFabricEntry

Max-Access not-accessible

Status obsolete

Description In some previous software releases, this table contained fabric information about the port to fabric name mapping of a given FC switch.

This table is no longer supported in current releases.

Sequence ::= { nishanEportDeviceInfo 3 }

---

eportSwitchFabricEntry

Syntax EportSwitchFabricEntry

Max-Access not-accessible

Status	obsolete
Description	Entry containing information for a Fabric Port Name to physical port mapping. INDEX { eportSwitchFabricFCSwitchName, eportSwitchFabricPort }
Sequence	::= { eportSwitchFabricTable 1 } EportSwitchFabricEntry ::= SEQUENCE { eportSwitchFabricFCSwitchName WWNtype, eportSwitchFabricPort INTEGER, eportSwitchFabricPortName WWNtype}

eportSwitchFabricFCSwitchName

Syntax	WWNtype
Max-Access	read-only
Status	obsolete
Description	The name of the FC switch which a given Fabric Port Name resides off.
Sequence	::= { eportSwitchFabricEntry 1 }

eportSwitchFabricPort

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	obsolete
Description	The physical port number on the switch which the fabric port name describes.
Sequence	::= { eportSwitchFabricEntry 2 }

eportSwitchFabricPortName

Syntax	WWNtype
Max-Access	read-only
Status	obsolete

Description The fabric port name of the physical port.

Sequence `::= { eportSwitchFabricEntry 3 }`

---

eportLunDiscoveryTable

Syntax SEQUENCE OF EportLunDiscoveryEntry

Max-Access not-accessible

Status obsolete

Description This table contains the Lun Discovery controls for a local R_Port. This command should only be performed on R_Ports that are principal. This table is not currently supported.

Sequence `::= { nishanEportDeviceInfo 4 }`

---

eportLunDiscoveryEntry

Syntax EportLunDiscoveryEntry

Max-Access not-accessible

Status obsolete

Description Entry containing R_Port discovery control information.

INDEX { eportLunDiscoveryPort }

Sequence `::= { eportLunDiscoveryTable 1 }`

EportLunDiscoveryEntry ::= SEQUENCE {

eportLunDiscoveryPort	INTEGER,
eportLunDiscoveryPortName	WWNtype,
eportLunDiscoveryStatus	INTEGER}

---

eportLunDiscoveryPort

Syntax INTEGER (1..4096)

Max-Access read-only

Status obsolete

Description The local eport of which to issue the LUN discovery request. This R_Port must be a principal R_Port.

Sequence `::= { eportLunDiscoveryEntry 1 }`

eportLunDiscoveryPortName

Syntax	WWNtype
Max-Access	read-write
Status	obsolete
Description	The device Port World Wide Name which is to be probed for its lun information off of a given R_Port subfabric. This is only writable if the eportLunDiscoveryStatus is not in progress.
Sequence	`::= { eportLunDiscoveryEntry 2 }`

eportLunDiscoveryStatus

Syntax	INTEGER { idle(1), inProgress(2), notReady(3), plogiFailed(4), prliFailed(5), scsiFailed(6), zoneFailed(7), notTarget(8), timeout(9), successful(10), aborted(11) }
Max-Access	read-only
Status	obsolete
Description	The status of the current lun discovery for a given R_Port.
Sequence	`::= { eportLunDiscoveryEntry 3 }`

eportDeviceDiscoveryTable

Syntax	SEQUENCE OF eportDeviceDiscoveryEntry
Max-Access	not-accessible

Status	current
Description	This table contains information about all the devices discovered on a given fabric.
Sequence	`::= { nishanEportDeviceInfo 5 }`

eportDeviceDiscoveryEntry

Syntax	EportDeviceDiscoveryEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular device discovered off of a given subfabric.
	INDEX { eportDeviceDiscoverySubFabric, eportDeviceDiscoveryPortName }
Sequence	`::= { eportDeviceDiscoveryTable 1 }`
	EportDeviceDiscoveryEntry ::= SEQUENCE {
	eportDeviceDiscoverySubFabric INTEGER,
	eportDeviceDiscoveryPortName WWNtype,
	eportDeviceDiscoveryRealFCID FCIDtype }

eportDeviceDiscoverySubFabric

Syntax	INTEGER (1..16)
Max-Access	read-only
Status	current
Description	The subfabric index associated with this discovered device.
Sequence	`::= { eportDeviceDiscoveryEntry 1 }`

eportDeviceDiscoveryPortName

Syntax	WWNtype
Max-Access	read-only
Status	current

Description The World Wide Port Name that uniquely identifies this discovered device.

Sequence `::= { eportDeviceDiscoveryEntry 2 }`

---

eportDeviceDiscoveryRealFCID

Syntax FCIDtype

Max-Access read-only

Status current

Description The FCID associated with this discovered device.

Sequence `::= { eportDeviceDiscoveryEntry 3 }`

---

eportDeviceImportMaxEntries

Syntax INTEGER (1..4096)

Max-Access read-only

Status current

Description The maximum number of devices that can be added to the import list.

Sequence `::= { nishanEportDeviceInfo 6 }`

---

eportDeviceImportCount

Syntax INTEGER (1..4096)

Max-Access read-only

Status current

Description The number of devices currently in the Import List.

Sequence `::= { nishanEportDeviceInfo 7 }`

---

eportDeviceImportTable

Syntax SEQUENCE OF eportDeviceImportEntry

Max-Access not-accessible

Status current

Description This table contains information about all the devices that is to be imported into the SAN Router fabric.

Sequence `::= { nishanEportDeviceInfo 8 }`

---

eportDeviceImportEntry

Syntax EportDeviceImportEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular device discovered off of a given subfabric.

INDEX { eportDeviceImportPortName }

Sequence `::= { eportDeviceImportTable 1 }`

EportDeviceImportEntry ::= SEQUENCE {

eportDeviceImportPortName	WWNtype,
eportDeviceImportRowStatus	RowStatus }

---

eportDeviceImportPortName

Syntax WWNtype

Max-Access read-only

Status current

Description The World Wide Port Name that uniquely identifies the device to import.

Sequence `::= { eportDeviceImportEntry 1 }`

---

eportDeviceImportRowStatus

Syntax RowStatus

Max-Access read-create

Status current

Description The row status of this import entry.

Sequence `::= { eportDeviceImportEntry 2 }`

R_Port Zoning Information

The following section of the MIB displays R_Port zoning information. The following table shows the current Zoning configuration found on a particular SAN island.

nishanEportZoning OBJECT IDENTIFIER ::= { nishanEport 2 }

eportZoneSetTable

Syntax	SEQUENCE OF EportZoneSetEntry
Max-Access	not-accessible
Status	current
Description	This table contains all the Zone Set for a FC SAN on a local R_Port.
Sequence	`::= { nishanEportZoning 1 }`

eportZoneSetEntry

Syntax	EportZoneSetEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular Zone Set off of a local R_Port. INDEX { eportZoneSetSoIPPort, eportZoneSetIndex }

Sequence	`::= { eportZoneSetTable 1 }`
	EportZoneSetEntry ::= SEQUENCE {
	eportZoneSetSoIPPort WWNtype,
	eportZoneSetIndex INTEGER,
	eportZoneSetName DisplayString,
	eportZoneSetIsActive INTEGER }

eportZoneSetSoIPPort

Syntax	WWNtype
Max-Access	read-only

Status	current
Description	The logical local R_Port off of which this particular configuration resides.
Sequence	`::= { eportZoneSetEntry 1 }`

eportZoneSetIndex

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	An arbitrary index which uniquely identifies a local Zone Set on this given port.
Sequence	`::= { eportZoneSetEntry 2 }`

eportZoneSetName

Syntax	DisplayString (SIZE (1..255))
Max-Access	read-only
Status	current
Description	The unique ASCII string that represents a Zone Set.
Sequence	`::= { eportZoneSetEntry 3 }`

eportZoneSetIsActive

Syntax	INTEGER { false(0), true(1) }
Max-Access	read-only
Status	current
Description	If this is the active zone set the value will be true otherwise false.
Sequence	`::= { eportZoneSetEntry 4 }`

eportZoneObjectTable

Syntax	SEQUENCE OF eportZoneObjectEntry
Max-Access	not-accessible

Status current

Description This table contains all the Zone Objects for a FC SAN on a local R_Port.

Sequence `::= { nishanEportZoning 2 }`

---

eportZoneObjectEntry

Syntax EportZoneObjectEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular Zone Object off of a local R_Port.

INDEX { eportZoneObjectSoIPPort, eportZoneObjectIndex }

Sequence `::= { eportZoneObjectTable 1 }`

EportZoneObjectEntry ::= SEQUENCE {

eportZoneObjectSoIPPort	WWNtype,
eportZoneObjectIndex	INTEGER,
eportZoneObjectName	DisplayString }

---

eportZoneObjectSoIPPort

Syntax WWNtype

Max-Access read-only

Status current

Description The logical local R_Port of which this particular Zone Object configuration resides.

Sequence `::= { eportZoneObjectEntry 1 }`

---

eportZoneObjectIndex

Syntax INTEGER (1..65535)

Max-Access read-only

Status current

Description An arbitrary index which uniquely identifies a local Zone Object on this given port.

Sequence `::= { eportZoneObjectEntry 2}`

---

eportZoneObjectName

Syntax DisplayString (SIZE (1..255))

Max-Access read-only

Status current

Description The unique ASCII string that represents a Zone Object.

Sequence `::= { eportZoneObjectEntry 3 }`

---

eportZoneAliasTable

Syntax SEQUENCE OF eportZoneAliasEntry

Max-Access not-accessible

Status current

Description This table contains all the zone aliases for a FC SAN on a local R_Port.

Sequence `::= { nishanEportZoning 3 }`

---

eportZoneAliasEntry

Syntax eportZoneAliasEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular Zone Set off of a local R_Port.

INDEX { eportZoneAliasSoIPPort, eportZoneAliasIndex }

Sequence `::= { eportZoneAliasTable 1 }`

EportZoneAliasEntry ::= SEQUENCE {

eportZoneAliasSoIPPort	WWNtype,
eportZoneAliasIndex	INTEGER,
eportZoneAliasName	DisplayString }

eportZoneAliasSoIPPort

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The logical local R_Port off of which this particular zone alias configuration resides.
Sequence	`::= { eportZoneAliasEntry 1 }`

eportZoneAliasIndex

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	An arbitrary index which uniquely identifies a local Zone Alias on this given port.
Sequence	`::= { eportZoneAliasEntry 2}`

eportZoneAliasName

Syntax	DisplayString (SIZE(1..255))
Max-Access	read-only
Status	current
Description	The unique ASCII string that represents a Zone Alias off of this given R_Port.
Sequence	`::= { eportZoneAliasEntry 3 }`

eportZoneSetToZoneTable

Syntax	SEQUENCE OF eportZoneSetToZoneTable
Max-Access	not-accessible
Status	current
Description	This table defines the relationship between a zone set object, and a zone object. The relationship defines whether or not a particular zone object belongs to a given zone set.

Sequence ::= { nishanEportZoning 4 }

eportZoneSetToZoneEntry

Syntax eportZoneSetToZoneEntry

Max-Access not-accessible

Status current

Description Entry containing which shows whether a particular zone object is part of a zone set.

INDEX { eportZoneSetToZoneSoIPPort, eportZoneSetToZoneSetIndex, eportZoneSetToZoneZoneIndex }

Sequence ::= { eportZoneSetToZoneTable 1 }

EportZoneSetToZoneEntry ::= SEQUENCE {

eportZoneSetToZoneSoIPPort WWNtype,

eportZoneSetToZoneSetIndex INTEGER,

eportZoneSetToZoneZoneIndex INTEGER }

eportZoneSetToZoneSoIPPort

Syntax WWNtype

Max-Access read-only

Status current

Description The logical local R_Port off of which this particular zone set and zone configuration resides.

Sequence ::= { eportZoneSetToZoneEntry 1 }

eportZoneSetToZoneSetIndex

Syntax INTEGER (1..65535)

Max-Access read-only

Status current

Description The zone set index defined in this particular relationship. This integer indexes into the eportZoneSetTable

Sequence ::= { eportZoneSetToZoneEntry 2 }

eportZoneSetToZoneZoneIndex

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	The zone object index defined in this particular relationship. This integer indexes into the eportZoneObjectTable.
Sequence	`::= { eportZoneSetToZoneEntry 3 }`

eportZoneObjectToWWNTable

Syntax	SEQUENCE OF eportZoneObjectToWWNEntry
Max-Access	not-accessible
Status	current
Description	This table contains the relationship of a zone, with a particular WWN. If an entry exists, the relationship defines that the particular WWN is a member of the zone.
Sequence	`::= { nishanEportZoning 5 }`

eportZoneObjectToWWNEntry

Syntax	EportZoneObjectToWWNEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular zone object and a given member off of a local R_Port. INDEX { eportZoneObjectToWWNSoIPPort, eportZoneObjectToWWNZoneIndex, eportZoneObjectToWWNDeviceName }
Sequence	`::= { eportZoneObjectToWWNTable 1 }`

```
EportZoneObjectToWWNEntry ::= SEQUENCE {
eportZoneObjectToWWNSoIPPort        WWNtype,
eportZoneObjectToWWNZoneIndex       INTEGER,
eportZoneObjectToWWNDeviceName      WWNtype }
```

---

eportZoneObjectToWWNSoIPPort

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This object defines the local port off of which this configuration resides.
Sequence	`::= { eportZoneObjectToWWNEntry 1 }`

---

eportZoneObjectToWWNZoneIndex

Syntax	INTEGER(1..65535)
Max-Access	read-only
Status	current
Description	This object defines the particular zone that a particular member belongs too. This integer indexes into the eportZoneObjectTable.
Sequence	`::= { eportZoneObjectToWWNEntry 2 }`

---

eportZoneObjectToWWNDeviceName

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This object defines the device that belongs to the zone in this entry.
Sequence	`::= { eportZoneObjectToWWNEntry 3 }`

---

eportZoneObjectToAliasTable

Syntax	SEQUENCE OF eportZoneObjectToAliasEntry

Max-Access not-accessible

Status current

Description This table contains the relationship of a zone, with a particular Alias. If an entry exists, the relationship defines that the particular Alias is a member of the Zone.

Sequence `::= { nishanEportZoning 6 }`

eportZoneObjectToAliasEntry

Syntax EportZoneObjectToAliasEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular zone object and a given alias off of a local R_Port.

INDEX { eportZoneObjectToAliasSoIPPort,
eportZoneObjectToAliasZoneIndex,
eportZoneObjectToAliasAliasIndex }

Sequence `::= { eportZoneObjectToAliasTable 1 }`

EportZoneObjectToAliasEntry ::= SEQUENCE {

eportZoneObjectToAliasSoIPPort	WWNtype,
eportZoneObjectToAliasZoneIndex	INTEGER,
eportZoneObjectToAliasAliasIndex	INTEGER}

eportZoneObjectToAliasSoIPPort

Syntax WWNtype

Max-Access read-only

Status current

Description This object defines the local port off of which this configuration resides. This integer indexes into the eportZoneAliasTable

Sequence `::= { eportZoneObjectToAliasEntry 1 }`

eportZoneObjectToAliasZoneIndex

Syntax	INTEGER(1..65535)
Max-Access	read-only
Status	current
Description	This object defines the particular zone that a particular member belongs too. This integer indexes into the eportZoneObjectTable.
Sequence	`::= { eportZoneObjectToAliasEntry 2 }`

eportZoneObjectToAliasAliasIndex

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	This object defines Zone Alias that belongs to the zone in this entry.
Sequence	`::= { eportZoneObjectToAliasEntry 3 }`

eportZoneObjectToPortZoneTable

Syntax	SEQUENCE OF eportZoneObjectToPortZoneEntry
Max-Access	not-accessible
Status	current
Description	This table contains the relationship of a zone, with a particular port zone. If an entry exists, the relationship defines that the particular domain, port combination is a member of the zone.
Sequence	`::= { nishanEportZoning 7 }`

eportZoneObjectToPortZoneEntry

Syntax	EportZoneObjectToPortZoneEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular zone object and a given member off of a local R_Port.

INDEX { eportZoneObjectToPortZoneSoIPPort,
eportZoneObjectToPortZoneZoneIndex,
eportZoneObjectToPortZoneDomain,
eportZoneObjectToPortZonePort }

Sequence `::= { eportZoneObjectToPortZoneTable 1 }`

EportZoneObjectToPortZoneEntry ::= SEQUENCE {

eportZoneObjectToPortZoneSoIPPort	WWNtype,
eportZoneObjectToPortZoneZoneIndex	INTEGER,
eportZoneObjectToPortZoneDomain	INTEGER,
eportZoneObjectToPortZonePort	INTEGER }

---

eportZoneObjectToPortZoneSoIPPort

Syntax WWNtype

Max-Access read-only

Status current

Description This object defines the local port off of which this configuration resides.

Sequence `::= { eportZoneObjectToPortZoneEntry 1 }`

---

eportZoneObjectToPortZoneZoneIndex

Syntax INTEGER (1..65535)

Max-Access read-only

Status current

Description This object defines the particular zone that a particular member belongs too. This integer indexes into the eportZoneObjectTable.

Sequence `::= { eportZoneObjectToPortZoneEntry 2 }`

---

eportZoneObjectToPortZoneDomain

Syntax INTEGER (1..240)

Max-Access read-only

Status	current
Description	This object defines the particular domain of a port zone.
Sequence	`::= { eportZoneObjectToPortZoneEntry 3 }`

eportZoneObjectToPortZonePort

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	This object defines the particular port of a port zone.
Sequence	`::= { eportZoneObjectToPortZoneEntry 4 }`

eportZoneAliasToWWNTable

Syntax	SEQUENCE OF eportZoneAliasToWWNEntry
Max-Access	not-accessible
Status	current
Description	This table contains the relationship of an alias, with a particular WWN. If an entry exists, the relationship defines that the particular WWN is a member of the alias.
Sequence	`::= { nishanEportZoning 8 }`

eportZoneAliasToWWNEntry

Syntax	EportZoneAliasToWWNEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular alias object and a given member off of a local R_Port. INDEX { eportZoneAliasToWWNSoIPPort, eportZoneAliasToWWNAliasIndex, eportZoneAliasToWWNDeviceName }
Sequence	`::= { eportZoneAliasToWWNTable 1 }`

EportZoneAliasToWWNEntry ::= SEQUENCE {

eportZoneAliasToWWNSoIPPort	WWNtype,
eportZoneAliasToWWNAliasIndex	INTEGER,
eportZoneAliasToWWNDeviceName	WWNtype }

eportZoneAliasToWWNSoIPPort

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This object defines the local port off of which this configuration resides.
Sequence	`::= { eportZoneAliasToWWNEntry 1 }`

eportZoneAliasToWWNAliasIndex

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	This object defines the particular alias that a particular member belongs too. This integer indexes into the eportZoneAliasTable.
Sequence	`::= { eportZoneAliasToWWNEntry 2 }`

eportZoneAliasToWWNDeviceName

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This object defines a device that belongs to a particular alias in this entry.
Sequence	`::= { eportZoneAliasToWWNEntry 3 }`

eportZoneAliasToPortZoneTable

Syntax	SEQUENCE OF eportZoneAliasToPortZoneEntry

Max-Access not-accessible

Status current

Description This table contains information about port zone members for a given alias.

Sequence `::= { nishanEportZoning 9 }`

---

eportZoneAliasToPortZoneEntry

Syntax EportZoneAliasToPortZoneEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular zone alias and a given member off of a local R_Port.

INDEX { eportZoneAliasToPortZoneSoIPPort,
eportZoneAliasToPortZoneAliasIndex,
eportZoneAliasToPortZoneDomain,
eportZoneAliasToPortZonePort }

Sequence `::= { eportZoneAliasToPortZoneTable 1 }`

EportZoneAliasToPortZoneEntry ::= SEQUENCE {

eportZoneAliasToPortZoneSoIPPort	WWNtype,
eportZoneAliasToPortZoneAliasIndex	INTEGER,
eportZoneAliasToPortZoneDomain	INTEGER,
eportZoneAliasToPortZonePort	INTEGER }

---

eportZoneAliasToPortZoneSoIPPort

Syntax WWNtype

Max-Access read-only

Status current

Description This object defines the local port off of which this configuration resides.

Sequence `::= { eportZoneAliasToPortZoneEntry 1 }`

eportZoneAliasToPortZoneAliasIndex

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	This object defines the particular alias that a particular member belongs to. This integer indexes into the eportZoneAliasTable.
Sequence	`::= { eportZoneAliasToPortZoneEntry 2 }`

eportZoneAliasToPortZoneDomain

Syntax	INTEGER (1..240)
Max-Access	read-only
Status	current
Description	This object defines the particular domain of a port zone.
Sequence	`::= { eportZoneAliasToPortZoneEntry 3 }`

eportZoneAliasToPortZonePort

Syntax	INTEGER (1..65535)
Max-Access	read-only
Status	current
Description	This object defines the particular port of a port zone.
Sequence	`::= { eportZoneAliasToPortZoneEntry 4 }`

R_Port Security Tables

The following section of the MIB displays R_Port security information. The following table shows the current security configuration found on a particular SAN island.

nishanEportSecurity OBJECT IDENTIFIER ::= { nishanEport 3 }

eportFabricBindingTotalCount

Syntax	INTEGER (1..4096)

Max-Access	read-only
Status	current
Description	This returns the count of the total number of entries in this table.
Sequence	`::= { nishanEportSecurity 1 }`

eportFabricBindingNextFreeIndex

Syntax	INTEGER (1..4096)
Max-Access	read-only
Status	current
Description	This returns the next free available index in this table that is free for use.
Sequence	`::= { nishanEportSecurity 2 }`

eportFabricBindingTable

Syntax	SEQUENCE OF EportFabricBindingEntry
Max-Access	not-accessible
Status	current
Description	This table comtains information on the fabric binding information that pertains to a given R_Port.
Sequence	`::= { nishanEportSecurity 3 }`

eportFabricBindingEntry

Syntax	EportFabricBindingEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular fabric binding entry. INDEX { eportFabricBindingIndex }
Sequence	`::= { eportFabricBindingTable 1 }`

EportFabricBindingEntry ::= SEQUENCE {

eportFabricBindingIndex	INTEGER,
eportFabricBindingSubFabric	INTEGER,
eportFabricBindingDomainID	INTEGER,
eportFabricBindingSwitchWWN	WWNtype,
eportFabricBindingRowStatus	RowStatus }

eportFabricBindingIndex

Syntax	INTEGER (1..4096)
Max-Access	not-accessible
Status	current
Description	A logcal index representing this fabric binding row.
Sequence	`::= { eportFabricBindingEntry 1 }`

eportFabricBindingSubFabric

Syntax	INTEGER (1..4096)
Max-Access	read-create
Status	current
Description	DURABLE: { 0: all } The logical local subfabric off which this particular fabricBinding resides.
Sequence	`::= { eportFabricBindingEntry 2 }`

eportFabricBindingDomainID

Syntax	INTEGER (1..255)
Max-Access	read-create
Status	current
Description	DURABLE: { 0 : all } A domain ID that belongs to a switchNodeName on this given subFabric.

Sequence `::= { eportFabricBindingEntry 3 }`

---

eportFabricBindingSwitchWWN

Syntax WWNtype

Max-Access read-create

Status current

Description DURABLE: { 0 : all }

The switchNode name that is assigned a unique domain on this given subFabric.

Sequence `::= { eportFabricBindingEntry 4 }`

---

eportFabricBindingRowStatus

Syntax RowStatus

Max-Access read-create

Status current

Description DURABLE: { 2 : all }

The status of a single fabric binding entry.

Sequence `::= { eportFabricBindingEntry 5 }`

---

eportSwitchBindingTotalCount

Syntax INTEGER (1..4096)

Max-Access read-only

Status current

Description This returns the count of the total number of entries in this table.

Sequence `::= { nishanEportSecurity 4 }`

---

eportSwitchBindingNextFreeIndex

Syntax INTEGER (1..4096)

Max-Access read-only

Status current

Description This returns the next free available index in this table that is free for use.

Sequence `::= { nishanEportSecurity 5 }`

eportSwitchBindingTable

Syntax SEQUENCE OF EportSwitchBindingEntry

Max-Access not-accessible

Status current

Description This table comtains information on the switch binding information that pertains to a given subfabric.

Sequence `::= { nishanEportSecurity 6 }`

eportSwitchBindingEntry

Syntax EportSwitchBindingEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular switch binding entry.

INDEX { eportSwitchBindingIndex }

Sequence `::= { eportSwitchBindingTable 1 }`

EportSwitchBindingEntry ::= SEQUENCE {

eportSwitchBindingIndex	INTEGER,
eportSwitchBindingSubFabric	INTEGER,
eportSwitchBindingSwitchWWN	WWNtype,
eportSwitchBindingRowStatus	RowStatus }

eportSwitchBindingIndex

Syntax INTEGER (1..4096)

Max-Access not-accessible

Status current

Description A logical index representing this Switch Binding Index binding row.

Sequence `::= { eportSwitchBindingEntry 1 }`

---

eportSwitchBindingSubFabric

Syntax INTEGER (1..4096)

Max-Access read-create

Status current

Description DURABLE: { 0 : all }

The logical subfabric off which this particular Switch Binding resides.

Sequence `::= { eportSwitchBindingEntry 2 }`

---

eportSwitchBindingSwitchWWN

Syntax WWNtype

Max-Access read-create

Status current

Description DURABLE: { 0 : all }

The switchNode name of the switch that this subfabric can be bound too.

Sequence `::= { eportSwitchBindingEntry 3 }`

---

eportSwitchBindingRowStatus

Syntax RowStatus

Max-Access read-create

Status current

Description DURABLE: { 2 : all }

The status of a signle switch binding entry.

Sequence `::= { eportSwitchBindingEntry 4 }`

---

eportPortBindingTable

Syntax SEQUENCE OF EportPortBindingEntry

Max-Access not-accessible

Status current

Description This table comtains information on the port binding information that pertains to a given R_Port.

Sequence `::= { nishanEportSecurity 7 }`

eportPortBindingEntry

Syntax EportPortBindingEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular port binding entry. Any changes to any entry requires the port offline followed by a port online.

INDEX { eportPortBindingEport }

Sequence `::= { eportPortBindingTable 1 }`

EportPortBindingEntry ::= SEQUENCE {

eportPortBindingEport	INTEGER,
eportPortBindingSwitchWWN	WWNtype,
eportPortBindingEnable	INTEGER,
eportPortBindingAttachedWWN	WWNtype }

eportPortBindingEport

Syntax INTEGER (1..4096)

Max-Access read-only

Status current

Description DURABLE: { 0 : all }

The physical local R_Port off which this particular Port Binding resides.

Sequence `::= { eportPortBindingEntry 1 }`

eportPortBindingSwitchWWN

Syntax WWNtype

Max-Access	read-write
Status	current
Description	DURABLE: { 0 : all } The immediately connected switchNode name that this R_Port can communicate.
Sequence	`::= { eportPortBindingEntry 2 }`

eportPortBindingEnable

Syntax	INTEGER {enabled(1),disabled(2) }
Max-Access	read-write
Status	current
Description	This enables port binidng for the local R_Port.
Sequence	`::= { eportPortBindingEntry 3 }`

eportPortBindingAttachedWWN

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The WWN node name of the immediately attached switch off this local R_Port.
Sequence	`::= { eportPortBindingEntry 4 }`

R_Port Route Tables

nishanEportRoute OBJECT IDENTIFIER ::= { nishanEport 4 }

eportPreferredPathCount

Syntax	INTEGER (1..4096)
Max-Access	read-only
Status	current
Description	This returns the count of the total number of entries in this table.

Sequence `::= { nishanEportRoute 1 }`

eporPreferredPathNextFreeIndex

Syntax INTEGER (1..4096)

Max-Access read-only

Status current

Description This returns the next free available index in this table that is free for use.

Sequence `::= { nishanEportRoute 2 }`

eportPreferredPathTable

Syntax SEQUENCE OF EportPreferredPathEntry

Max-Access not-accessible

Status current

Description This table contains information on the static route information that pertains to a given R_Port.

Sequence `::= { nishanEportRoute 3 }`

eportPreferredPathEntry

Syntax EportPreferredPathEntry

Max-Access not-accessible

Status current

Description Entry containing information for a particular R_Port static route.

INDEX { eportPreferredPathIndex }

Sequence `::= { eportPreferredPathTable 1 }`

EportPreferredPathEntry ::= SEQUENCE {

eportPreferredPathIndex	INTEGER,
eportPreferredPathInPort	INTEGER,
eportPreferredPathOutEPort	INTEGER,
eportPreferredPathRowStatus	RowStatus }

eportPreferredPathIndex

Syntax	INTEGER(1..4096)
Max-Access	not-accessible
Status	current
Description	This is the logical index of this particular preferred path entry.
Sequence	`::= { eportPreferredPathEntry 1 }`

eportPreferredPathInPort

Syntax	INTEGER(1..4096)
Max-Access	read-create
Status	current
Description	The physical IN-Port that this preferred path entry applies too. This can be any portType E-port, F-port, GigEPort.
Sequence	`::= { eportPreferredPathEntry 2 }`

eportPreferredPathOutEPort

Syntax	INTEGER(1..4096)
Max-Access	read-create
Status	current
Description	The physical Out-Port that this preferred path entry applies too. This must be an R_Port that which directly connects to the subfabric this entry represents.
Sequence	`::= { eportPreferredPathEntry 3 }`

eportPreferredPathRowStatus

Syntax	RowStatus
Max-Access	read-create
Status	current
Description	The status of this preferred path entry.
Sequence	`::= { eportPreferredPathEntry 4 }`

nishanClusterID

Syntax	INTEGER (1..63)
Max-Access	read-write
Status	current
Description	The cluster ID uniquely identifies a unique collection of gateways across multiple IPS, and FC Fabric installations. Those gateways interconnected via mFCP must all have the same cluster ID.

NOTE: Any changes to the cluster ID requires all R_Ports to go offline followed by an online.

Sequence `::= { nishanEportRoute 4 }`

R_Port Subfabric Tables

nishanEportSubFabric OBJECT IDENTIFIER ::= { nishanEport 5 }

eportSubFabricConfigurationTable

Syntax	SEQUENCE OF EportSubFabricEntry
Max-Access	not-accessible
Status	current
Description	This table contains configuration information for a given subfabric.
Sequence	`::= { nishanEportSubFabric 1}`

eportSubFabricEntry

Syntax	EportSubFabricEntry
Max-Access	not-accessible
Status	current
Description	Entry containing information for a particular subfabric entry. INDEX { eportSubFabricIndex }
Sequence	`::= { eportSubFabricConfigurationTable 1 }`

```
EportSubFabricEntry ::= SEQUENCE {
eportSubFabricIndex                     INTEGER,
eportSubFabricInterconnect              INTEGER,
eportSubFabricZoneSetPolicy             INTEGER,
eportSubFabricCollectFcSwitchInfo       INTEGER,
eportSubFabricFabricBinding             INTEGER,
eportSubFabricSwitchBinding             INTEGER,
eportSubFabricEnterpriseFabricMode      INTEGER,
eportSubFabricName                      OCTET STRING,
eportSubFabricPrincipalSwitch           WWNtype,
eportSubFabricFcFabricProxyWWN          WWNtype,
eportSubFabricLocalProxyWWN             WWNtype,
eportSubFabricDiscoveredDeviceCount     INTEGER }
```

eportSubFabricIndex

Syntax	INTEGER(1..4096)
Max-Access	not-accessible
Status	current
Description	The Logical Index representing this subfabric entry.
Sequence	`::= { eportSubFabricEntry 1}`

eportSubFabricInterconnect

Syntax	INTEGER { openFabric(1), mcDataNative(2), brocade(3) }
Max-Access	read-write
Status	current
Description	DURABLE: {2:all} This object shows what interoperability mode a SubFabric operates in. Note if running in openFabric or mcDataNative mode Domain Id's are limited to a value between hex 1..1F. Changes to this entry requires the port to go offline followed by an online. DEFVAL {openFabric}

Sequence ::= { eportSubFabricEntry 2}

eportSubFabricZoneSetPolicy

Syntax INTEGER { noZoneSynch(1), appendIPSZones(2), createIPSZoneset(3) }

Max-Access read-write

Status current

Description DURABLE: { leaveCurrentZoneSet:all }

This object determines the action that is taken by Nishan E-Port with the connected FC SAN island. If set to createIPSZoneset, this switch will always force the SoIPZoneset on a given island to be the active configuration, otherwise if set to leaveCurrentZoneset the current active ZoneSet is not altered. If set to appendIPSZones, the SoIP zones shall be appended to the active FC zoneset on the FC Island. If there is no active FC zoneset, no append shall occur. Changes to this entry requires the port to go offline followed by an online.

Sequence ::= { eportSubFabricEntry 3}

eportSubFabricCollectFcSwitchInfo

Syntax INTEGER { allowed(1), disallowed(2) }

Max-Access read-write

Status current

Description DURABLE: { disallowed:all }

Setting this variable will allow the E_Port to collect the port information of the connected switches in the subfabric. Changes to this entry requires the port to go offline followed by an online.

DEFVAL {disallowed}

Sequence ::= { eportSubFabricEntry 4 }

eportSubFabricFabricBinding

Syntax INTEGER { enabled(1), disabled(2) }

Max-Access read-write

Status current

Description DURABLE: { disabled:all }

This object is used by R_Port to determine whether Fabric Binding is enabled or not for this given SubFabric.

NOTE: If enterpriseFabricMode is enabled, a set to disable Fabric Binding will result in an error.

Changes to this entry requires the port to go offline followed by an online.

Sequence `::= { eportSubFabricEntry 5 }`

eportSubFabricSwitchBinding

Syntax INTEGER { enabled(1), disabled(2) }

Max-Access read-write

Status current

Description DURABLE: { disabled:all }

This object is used by R_Port to determine whether Switch Binding is enabled or not for this given port.

NOTE: If enterpriseFabricMode is enabled, a set to disable Switch Binding will result in an error. Changes to this entry requires the port to go offline followed by an online.

Sequence `::= { eportSubFabricEntry 6 }`

eportSubFabricEnterpriseFabricMode

Syntax INTEGER {enabled(1), disabled(2) }

Max-Access read-write

Status current

Description DURABLE: { disabled:all }

This object is used by R_Port to determine whether Enterprise Fabric Mode is enabled or not for this given port. Changes to this entry requires the port to go offline followed by an online.

Sequence `::= { eportSubFabricEntry 7 }`

eportSubFabricName

Syntax	OCTET STRING (SIZE (0..32))
Max-Access	read-write
Status	current
Description	This is the name of a given sub fabric, it is user editable.
Sequence	`::= { eportSubFabricEntry 8 }`

eportSubFabricPrincipalSwitch

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This variables shows the Switch Node name of the principal switch in this subfabric.
Sequence	`::= { eportSubFabricEntry 9 }`

eportSubFabricFcFabricProxyWWN

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This is the WWN for the FcProxy proxy domain of this given Sub Fabric.
Sequence	`::= { eportSubFabricEntry 10 }`

eportSubFabricLocalProxyWWN

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This is the WWN for the local proxy domain of this given Sub Fabric.
Sequence	`::= { eportSubFabricEntry 11 }`

eportSubFabricDiscoveredDeviceCount

Syntax	INTEGER(1..4096)
Max-Access	read-only
Status	current
Description	This is the count of active entries in the Device Discovery Table for this given Sub Fabric.
Sequence	`::= { eportSubFabricEntry 12 }`

eportSubFabricMaximumConfigurable

Syntax	INTEGER (1..4096)
Max-Access	read-only
Status	current
Description	This variable shows the maximum number of usable Subfabrics for this given product.
Sequence	`::= { nishanEportSubFabric 2 }`
	END

C

McDATA Gateway MIB

Nishan Gateway MIB (iFCP Gateway) V1

This MIB contains management objects for McDATA iFCP gateway support.

Revision history:

10/??/00	Initial version created.
4/24/01	Minor corrections to compile without warnings in SMICng.
5/11/01	Added DURABLE keyword for saturn objects to be saved to flash.
5/24/01	Added new variable localPeerAction, updated enumerated values for localPeerFailoverStatus and rmtPeerAction.
5/28/01	Changed allowed range for localPeerHoldTime and remotePeerDefaultHoldTime
6/12/01	Added localPeerFailoverTable
6/15/01	Corrected range on rmtPeerLocalGigeIndex
6/28/01	Changed allowed range for localPortHoldTime
6/29/01	Changed allowed range for localPortHoldTime again for testing
9/25/01	Changed default value for remotePeerDefautlHoldTimer and for localPortHoldTime

9/26/01 Changed allowed range for localPortHoldTime fixed indentation, again.

11/16/01 Added localPortBackupConnect

12/11/01 Changed allowed range for remotePeerDefaultHoldTime, rmtPeerHoldTime and lclPeerRemoteHoldTime to 10 - 90 seconds, default 30.

05/24/02 Clarified Remote Peer and Local Peer table descriptions. added lclPeerFailoverCount, lclPeerDiscoveredPathMtuSize, rmtPeerDiscoveredPathMtuSize, rmtPeerMaxMtuSize

06/14/02 Made rmtPeerMaxMtuSize read-only. Max MTU size is now set per port in the Nishan Mgt MIB

03/07/03 Added rmtPeerTcpWindowSize to rmtPeerTable, and added lclPeerTcpWindowSize to lclPeerRedundantSupportTable.

03/12/03 Included support for IPS 5000 series and 3350 models.

04/11/03 Added new rmtConnectionCount and rmtConnectionTable to replace rmtPeerTable.

04/17/03 Added rmtConnTcpTotalSlowStarts variable to count slow starts in iFCP remote peer connections.

04/27/03 Improved the description for rmtConnTcpTotalSlowStarts.

05/21/03 Update rmtConnConnectionStatus and lclPeerConnectionStatus descriptions.

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

NISHAN-GTWY DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE, IpAddress, Gauge32, Counter32, TimeTicks, Unsigned32

FROM SNMPv2-SMI

DisplayString, RowStatus

FROM SNMPv2-TC

nishanMgmt

FROM NISHAN-MGT;

nishanGtwyPeer

MODULE-IDENTITY

Last-Updated 0509190000Z

Organization McDATA Corporation

Contact-Info McDATA Corporation

380 Interlocken Crescent

Broomfield, CO 80021 USA

Tel : 1 720 558-8000

Fax : 1 720 558-3860

email : information@mcdata.com

Description The private MIB for McDATA Eclipse SAN Router Management.

Revision 0509190000Z

Description Released for E/OSi version 4.7.

Revision 0506160000Z

Description Multiple comment and description updates. Removed the localPeerFailoverTable, not supported in any current products.

Revision 0304270000Z

Description Public release of this MIB module, as of 4/27/03.

Sequence `::= { nishanMgmt 11 }`

Local SAN configuration

The following setup parameters are used to configure the SAN Router to manage the local SAN.

localSAN OBJECT IDENTIFIER ::= { nishanGtwyPeer 1 }

remotePeerDefaultHoldTime

Syntax INTEGER (10 .. 90)

Max-Access read-write

Status	current
Units	"seconds"
Description	DURABLE: { 30 } The hold time, in seconds, used by default between the local and remote peers. Valid values are 10 .. 90. DEFVAL {30}
Sequence	::= { localSAN 1 }

localSanId

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	The SAN ID of the local network. The format is a 4-Byte unsigned value.
Sequence	::= { localSAN 2 }

localSanIdOnNextReset

Syntax	Unsigned32
Max-Access	read-write
Status	current
Description	DURABLE: The SAN ID of the local network after system reset. The format is a 4-Byte unsigned value.
Sequence	::= { localSAN 3 }

localRecoveryAction

Syntax	INTEGER { manual(0), automatic(1) }
Max-Access	read-write
Status	obsolete
Description	This variable determined backup recovery operation on old discontinued products, and is no longer supported. On current products, the recovery action is configured separately for each port. See localPortRecoveryAction in the localPortFailoverTable. DEFVAL {manual}

Sequence ::= { localSAN 4 }

localRecoverNow

Syntax INTEGER { idle(0), recover(1) }

Max-Access read-write

Status obsolete

Description This variable initiated the backup recovery operation on old discontinued products, and is no longer supported. On current products, recovery is initiated separately for each port. See localPortRecoverNow in the localPortFailoverTable.

DEFVAL {idle}

Sequence ::= { localSAN 5 }

localPortFailoverTable

Syntax SEQUENCE OF LocalPortFailoverEntry

Max-Access not-accessible

Status current

Description This table configures an optional backup port for each TCP port. The backup port may be on the same switch or a different switch. Rows are not created or deleted in this table. There is one permanent row for each TCP port in the switch. The backup information for a TCP port remains even if the port is not currently configured for iFCP.

Sequence ::= { localSAN 7 }

localPortFailoverEntry

Syntax LocalPortFailoverEntry

Max-Access not-accessible

Status current

Description Configuration information for one port backup.

INDEX {localPortIndex} ::= { localPortFailoverTable 1 }

```
LocalPortFailoverEntry ::= SEQUENCE {
localPortIndex                  INTEGER,
localPortFailoverIpAddress      IpAddress,
localPortHoldTime               INTEGER,
localPortAction                 INTEGER,
localPortFailoverStatus         INTEGER,
localPortRecoveryAction         INTEGER,
localPortRecoverNow             INTEGER,
localPortBackupConnect          INTEGER
 }
```

localPortIndex

Syntax	INTEGER (1..16)
Max-Access	read-only
Status	current
Description	The number of the TCP port being backed up.
Sequence	`::= { localPortFailoverEntry 1 }`

localPortFailoverIpAddress

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	DURABLE: The internal proxy IP Address of the TCP port which is providing backup for this port. This may be the internal proxy address of a port on this same switch, or a different switch.
Sequence	`::= { localPortFailoverEntry 2 }`

localPortHoldTime

Syntax	INTEGER (5 .. 60)
Max-Access	read-write
Status	current

UNITS "seconds"

Description — DURABLE: { 5:all } The time-out value, in seconds, that the backup port should wait before declaring the primary port as down. If the primary port remains unreachable for this period of time, the backup port will take over the primary port's remote connections.

DEFVAL {5}

Sequence — `::= { localPortFailoverEntry 3 }`

localPortAction

Syntax — INTEGER { disable(0), enable(1) }

Max-Access — read-write

Status — current

Description — DURABLE: { disable:all } Administrative status for the port backup. If set to 1 (enable), the backup port takes over the primary port's connections if the hold time expires. If set to 0 (disable) the backup port does not monitor the primary port, and does not detect a primary port failure.

DEFVAL {disable}

Sequence — `::= { localPortFailoverEntry 4 }`

localPortFailoverStatus

Syntax — INTEGER { down(0), accepted(1), rejected(2) }

Max-Access — read-only

Status — current

Description — The status of the port backup configuration.

Down	the configured backup port cannot be reached, or the port backup operation is disabled.
Accepted	the specified backup port has accepted the fail-over request.
Rejected	the specified backup port has rejected the fail-over request.

DEFVAL {down}

Sequence — `::= { localPortFailoverEntry 5 }`

localPortRecoveryAction

Syntax	INTEGER { manual(0), automatic(1) }
Max-Access	read-write
Status	current
Description	DURABLE: { manual:all } Determines the recovery action when this port or this switch comes back up: Manual(0) — Connections currently handled by the backup port are left down for this port. The connections continue to be handled by the backup port. To transfer the connection back to this port, the user should use the localPortRecoverNow variable. Automatic(1) — Connections currently handled by the backup port are terminated in the backup switch. If the connection is enabled, this port re-opens the connection. The backed-up connections are transferred back to this port automatically. DEFVAL {manual}
Sequence	`::= { localPortFailoverEntry 6 }`

localPortRecoverNow

Syntax	INTEGER { idle(0), recover(1) }
Max-Access	read-write
Status	current
Description	DURABLE: { idle:all } When localPortRecoveryAction is set to Manual(0), localPortRecoverNow is used to trigger a manual recovery of connections previously transferred to a backup port. When read, localPortRecoverNow always returns 0. Setting localPortRecoverNow to 0 has no effect. Setting localPortRecoverNow to 1 causes this port to: - inform the backup port to terminate all active backup connections. - re-establish primary connections to replace the backup connections.

If there are no connections being backed up, or when localPortRecoveryAction is set to Automatic(1), setting localPortRecoverNow to 1 has no effect.

DEFVAL {idle}

Sequence `::= { localPortFailoverEntry 7 }`

localPortBackupConnect

Syntax INTEGER { primaryFirst(1), backupOnly(2) }

Max-Access read-write

Status current

Description DURABLE: { primaryFirst:all } The remote port that the backup port should connect to when the backup link is activated. If set to 'primary', the backup port attempts to connect to the remote primary address first. If that times out, the backup port attempts to connect to the remote backup address. If set to 'backup', the backup port attempts to connect only to the remote backup address. If the backup path is isolated from the primary path (no connectivity from the local backup port to the remote primary address), set this parameter to 'backupOnly' to recover from a link failure more quickly.

DEFVAL {primaryFirst}

Sequence `::= { localPortFailoverEntry 8 }`

iFCP Remote Peer Configuration

remotePeer OBJECT IDENTIFIER ::= { nishanGtwyPeer 2 }

NOTE: the rmtPeerTable below is used in switch software versions 4.0 and earlier. In 4.1 and later versions, the rmtPeerTable is replaced by the rmtConnectionTable.

rmtPeerTable

Syntax	SEQUENCE OF RmtPeerEntry
Max-Access	not-accessible
Status	deprecated
Description	The remote peer table contains the configured list of remote SAN Routers. The remote peer table provides the ability to export zones to a remote peer SAN Router. It also shows the current status of the exported zones.
Sequence	`::= { remotePeer 1 }`

rmtPeerEntry

Syntax	RmtPeerEntry
Max-Access	not-accessible
Status	deprecated
Description	The information relating to the specified remote peer. INDEX { rmtPeerPrimaryIPAddress }
Sequence	`::= { rmtPeerTable 1 }`

RmtPeerEntry ::= SEQUENCE {

rmtPeerPrimaryIPAddress	IpAddress,
rmtPeerFailoverIPAddress	IpAddress,
rmtPeerLocalGigeIndex	INTEGER,
rmtPeerHoldTime	INTEGER,
rmtPeerAction	INTEGER,
rmtPeerConnectionStatus	INTEGER,
rmtPeerConnectionDesc	DisplayString,

rmtPeerRmtSanID	Unsigned32,
rmtPeerRcvBytes	Counter32,
rmtPeerXmtBytes	Counter32,
rmtPeerConnections	Gauge32,
rmtPeerLatency	Gauge32,
rmtPeerLostConnections	Counter32,
rmtPeerLostPackets	Counter32,
rmtPeerTotalPackets	Counter32,
rmtPeerUpTime	TimeTicks,
rmtPeerDiscoveredPathMtuSize	INTEGER,
rmtPeerMaxMtuSize	INTEGER,
rmtPeerRowStatus	RowStatus,
rmtPeerTcpWindowSize	INTEGER
}	

rmtPeerPrimaryIPAddress

Syntax	IpAddress
Max-Access	read-only
Status	deprecated
Description	The external IP Address of the primary remote peer SAN Router that this entry corresponds to.
Sequence	`::= { rmtPeerEntry 1 }`

rmtPeerFailoverIPAddress

Syntax	IpAddress
Max-Access	read-only
Status	deprecated
Description	The IP Address of the failover remote peer SAN Router used to backup the primary peer SAN Router. If no backup has been configured then the entry is set to all 0's.
Sequence	`::= { rmtPeerEntry 2 }`

rmtPeerLocalGigeIndex

Syntax	INTEGER (1 .. 16)
Max-Access	read-create
Status	deprecated
Description	The Gigabit Ethernet port index being used for this connection on the switch.
Sequence	`::= { rmtPeerEntry 3 }`

rmtPeerHoldTime

Syntax	INTEGER (10 .. 90)
Max-Access	read-create
Status	deprecated
Description	The hold time, measured in seconds, used between the local and remote peers. DEFVAL {30}
Sequence	`::= { rmtPeerEntry 4 }`

rmtPeerExpZoneListBitMap

Syntax	OCTET STRING (SIZE (64))
Max-Access	read-create
Status	deprecated
Description	A bit map indicating the zones requested to be exported from the local peer to the remote peer indicated. Zone 1 is represented by the least significant bit (rightmost) of byte 0.
Sequence	`::= { rmtPeerEntry 5 }`

rmtPeerAction

Syntax	INTEGER { disable(0), enable(1) }
Max-Access	read-create
Status	deprecated

Description This connection is enabled by setting the remotePeerAction status to Enabled.

DEFVAL { disable }

Sequence `::= { rmtPeerEntry 6 }`

rmtPeerConnectionStatus

Syntax INTEGER { down(0), standby(1), primaryactive(2), failoveractive(3), backup(4) }

Max-Access read-only

Status deprecated

Description This indicates the current status of the connection.

down(0)	Connection is not operating.
standby(1)	This value is not used.
primaryactive(2)	This SAN Router is connected to the remote primary SAN Router.
failoveractive(3)	This SAN Router is connected to the remote backup SAN Router.
backup(4)	The connection is not operating on this SAN Router, but has been activated on our backup SAN Router.

Sequence `::= { rmtPeerEntry 7 }`

rmtPeerConnectionDesc

Syntax DisplayString (SIZE (0..32))

Max-Access read-create

Status deprecated

Description A string describing the connection. Used for management purposes.

DEFVAL { "" }

Sequence `::= { rmtPeerEntry 8 }`

rmtPeerRmtSanID

Syntax Unsigned32

Max-Access	read-only
Status	deprecated
Description	The SAN ID of the remote SAN. The format is a 4-Byte unsigned value. This is determined throught inter-SAN Router communication.
Sequence	`::= { rmtPeerEntry 9 }`

rmtPeerRcvBytes

Syntax	Counter32
Max-Access	read-only
Status	deprecated
Description	The total number of bytes received from the remote SAN Router since the connection was first established, including MAC headers and CRC.
Sequence	`::= { rmtPeerEntry 10 }`

rmtPeerXmtBytes

Syntax	Counter32
Max-Access	read-only
Status	deprecated
Description	The total number of bytes transmitted by this SAN Router since the connection was first established, including MAC headers and CRC.
Sequence	`::= { rmtPeerEntry 11 }`

rmtPeerConnections

Syntax	Gauge32
Max-Access	read-only
Status	deprecated
Description	The current number of initiator/target pair data connections, between the local and remote SAN Router, using the iFCP link. For example, if one local server uses four remote disks, four connections are counted.

Sequence ::= { rmtPeerEntry 12 }

rmtPeerLatency

Syntax Gauge32

Max-Access read-only

Status deprecated

Description The round trip time (RTT) between the local and remote SAN Routers, measured in milliseconds, of the most recent connection keep-alive message. The RTT measurement is updated at one-third the interval of the rmtPeerHoldTime of the connection.

Sequence ::= { rmtPeerEntry 13}

rmtPeerLostConnections

Syntax Counter32

Max-Access read-only

Status deprecated

Description The number of initiator/target pair data connections, between the local and remote SAN Router, that had to be abnormally terminated. The reason the connections were terminated may be because the hold time expired, the port link status went down, or other action that caused lost connectivity on the link.

Sequence ::= { rmtPeerEntry 14}

rmtPeerLostPackets

Syntax Counter32

Max-Access read-only

Status deprecated

Description The number of TCP segments sent by this SAN Router that had to be retransmitted.

Sequence ::= { rmtPeerEntry 15 }

rmtPeerTotalPackets

Syntax	Counter32
Max-Access	read-only
Status	deprecated
Description	The number of TCP segments transmitted by this SAN Router. This does not recount a segment that was retransmitted.
Sequence	`::= { rmtPeerEntry 16 }`

rmtPeerUpTime

Syntax	TimeTicks
Max-Access	read-only
Status	deprecated
Description	The time, in hundredths of a second, since the current control connection to the remote SAN Router was established. This is the length of time that the current connection to the remote SAN Router has been up and running. If the remote SAN Router connection is currently down or disabled, this variable should return 0.
Sequence	`::= { rmtPeerEntry 17 }`

rmtPeerDiscoveredPathMtuSize

Syntax	INTEGER (0 .. 65535)
Max-Access	read-only
Status	current
Description	The Discovered Path Maximum Transmission Unit (MTU) Size, in bytes, is the MTU size that was discovered for the path used by this connection. The Path MTU is determined at connection initiation, and when the connection is made aware of route changes. The actual MTU used by the connnection is either the Discovered Path MTU or rmtPeerMaxMtuSize, whichever is smaller. If the rmtPeerMaxMtuSize has not been set (indicated by a value of 0), then it is ignored.
Sequence	`::= { rmtPeerEntry 18 }`

rmtPeerMaxMtuSize

Syntax	INTEGER (0 .. 65535)
Max-Access	read-only
Status	current
Description	The maximum value, in bytes, that can be used for the Maximum Transmission Unit (MTU) Size for the connection. This variable reflects what has been set for the TCP port that is hosting this connection. The value for the port is set by using NISHAN-MGT fcswTCPMaxMtuSize. This variable can used to prevent accidental fragmentation when the Path MTU value, as returned by paths used by this connection, is incorrect causing accidental fragmentation. If this variable is set to 0, the default setting, then it is ignored.
Sequence	`::= { rmtPeerEntry 19 }`

rmtPeerRowStatus

Syntax	RowStatus
Max-Access	read-create
Status	current
Description	This indicates the status of this entry. The row status is for creating the row entry. It does not show the status of the connection. The connection status is determined by the Action and Status attributes.

active (1),	read-write
notInService (2),	read-write
notReady (3),	read-only
createAndGo (4),	write-only
createAndWait (5),	write-only
destroy (6),	write-only

Sequence	`::= { rmtPeerEntry 20 }`

rmtPeerTcpWindowSize

Syntax	INTEGER (0 .. 8184)
Max-Access	read-create

Status current

Description DURABLE: { 0:all } The TCP window size used for all TCP data connections in this iFCP remote connection, in KBytes. The value configured here is rounded up to the next multiple of 8 KBytes internally. The TCP window size specifies how much outstanding data (data not yet acknowledged by the receiver) may be sent on one TCP connection. The window size is also the receive buffer size for each TCP connection. The recommended TCP window size is the expected bandwidth multiplied by the round-trip time (RTT). For example, the optimal window size for a 100 Mbit/sec connection with a 5 msec RTT is 100Mbit/sec * .005 sec = 0.5Mbits or about 65 KBytes.

The value 0 is a special value meaning 'Auto'. In Auto mode, the switch selects the optimal receive window size for each TCP connection independently. The window size is based on the TCP stack measured round-trip times. Each connection is adjusted dynamically as the connection's RTT changes. In the case of oversubscription, buffering resources are distributed fairly across connections to the extent possible. In the automode, each connection has a minimum window size of 128 KB and a maximum window size of 8184 KB.

The TCP window size may be manually changed at any time by setting this MIB variable. The new value begins taking effect immediately. Large changes may take a few seconds to become fully effective, as the window size changes gradually. TCP window size changes do not interrupt the remote SAN Router traffic.
DEFVAL { 0 }

Sequence `::= { rmtPeerEntry 21 }`

---

rmtConnectionCount

Syntax INTEGER (0..2147483647)

Max-Access read-only

Status current

Description The current number of entries in rmtTable. This is a count of all configured remote connections, including disabled and down connections.

Sequence `::= { remotePeer 2 }`

Remote Connection Table

NOTE: The rmtConnectionTable below is available only in switch software version 4.1 or later. In earlier software versions, the rmtPeerTable is provided instead.

rmtConnectionTable

Syntax	SEQUENCE OF RmtConnectionEntry
Max-Access	not-accessible
Status	current
Description	The remote connection table contains the configured list of remote SAN Routers. The remote connection table provides the ability to export zones to a remote peer SAN Router.
Sequence	`::= { remotePeer 3 }`

rmtConnectionEntry

Syntax	RmtConnectionEntry
Max-Access	not-accessible
Status	current
Description	The information relating to the specified remote peer. INDEX { rmtConnLocalGigeIndex, rmtConnPrimaryIPAddress }
Sequence	`::= { rmtConnectionTable 1 }`

RmtConnectionEntry ::= SEQUENCE {

rmtConnLocalGigeIndex	INTEGER,
rmtConnPrimaryIPAddress	IpAddress,
rmtConnFailoverIPAddress	IpAddress,
rmtConnHoldTime	INTEGER,
rmtConnExpZoneListBitMap	OCTET STRING,
rmtConnAction	INTEGER,
rmtConnConnectionStatus	INTEGER,
rmtConnConnectionDesc	DisplayString,
rmtConnRmtSanID	Unsigned32,

rmtConnRcvBytes	Counter32,
rmtConnXmtBytes	Counter32,
rmtConnConnections	Gauge32,
rmtConnLatency	Gauge32,
rmtConnLostConnections	Counter32,
rmtConnLostPackets	Counter32,
rmtConnTotalPackets	Counter32,
rmtConnUpTime	TimeTicks,
rmtConnDiscoveredPathMtuSize	INTEGER,
rmtConnMaxMtuSize	INTEGER,
rmtConnTcpWindowSize	INTEGER,
rmtConnTcpTotalSlowStarts	Counter32,
rmtConnRowStatus	RowStatus }

---

rmtConnLocalGigeIndex

Syntax	INTEGER (1 .. 16)
Max-Access	read-only
Status	current
Description	The Gigabit Ethernet port index on the local switch being used for this connection.
Sequence	`::= { rmtConnectionEntry 1 }`

---

rmtConnPrimaryIPAddress

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The external IP Address of the TCP port at the remote endof this iFCP connection.
Sequence	`::= { rmtConnectionEntry 2 }`

rmtConnFailoverIPAddress

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The IP Address of the failover remote peer SAN Router used to backup the primary peer SAN Router. If no backup has been configured then the entry is set to all 0's.
Sequence	`::= { rmtConnectionEntry 3 }`

rmtConnHoldTime

Syntax	INTEGER (10 .. 90)
Max-Access	read-create
Status	current
Description	The hold time, measured in seconds, used between the local and remote peers. DEFVAL {30}
Sequence	`::= { rmtConnectionEntry 4 }`

rmtConnExpZoneListBitMap

Syntax	OCTET STRING (SIZE (64))
Max-Access	read-create
Status	current
Description	A bit map indicating the zones requested to be exported from the local peer to the remote peer indicated. Zone 1 is represented by the least significant bit (rightmost) of byte 0.
Sequence	`::= { rmtConnectionEntry 5 }`

rmtConnAction

Syntax	INTEGER { disable(0), enable(1) }
Max-Access	read-create
Status	current

Description This connection is enabled by setting the remotePeerAction status to Enabled.

DEFVAL { disable }

Sequence `::= { rmtConnectionEntry 6 }`

---

rmtConnConnectionStatus

Syntax INTEGER

{ down(0), standby(1), primaryactive(2), failoveractive(3), backup(4) }

Max-Access read-only

Status current

Description This indicates the current status of the connection.

down(0)	Connection is not operating.
standby(1)	This value is not used.
primaryactive(2)	The connection is up and running.
failoveractive(3)	Connection is not operating.
backup(4)	Connection is not operating.

The precise differences between values 0, 3, and 4 are not known.

Sequence `::= { rmtConnectionEntry 7 }`

---

rmtConnConnectionDesc

Syntax DisplayString (SIZE (0..32))

Max-Access read-create

Status current

Description A string describing the connection. Used for management purposes.

DEFVAL { "" }

Sequence `::= { rmtConnectionEntry 8 }`

---

rmtConnRmtSanID

Syntax Unsigned32

Max-Access read-only

Status current

Description The SAN ID of the remote SAN. The format is a 4-Byte unsigned value. This is determined through inter-SAN Router communication.

Sequence `::= { rmtConnectionEntry 9 }`

---

rmtConnRcvBytes

Syntax Counter32

Max-Access read-only

Status current

Description The total number of bytes received from the remote SAN Router since the connection was first established, including MAC headers and CRC.

Sequence `::= { rmtConnectionEntry 10 }`

---

rmtConnXmtBytes

Syntax Counter32

Max-Access read-only

Status current

Description The total number of bytes transmitted by this SAN Router since the connection was first established, including MAC headers and CRC.

Sequence `::= { rmtConnectionEntry 11 }`

---

rmtConnConnections

Syntax Gauge32

Max-Access read-only

Status current

Description The current number of initiator/target pair data connections,between the local and remote SAN Router, using the iFCP link. For example, if one local server uses four remote disks, four connections are counted.

Sequence `::= { rmtConnectionEntry 12 }`

---

rmtConnLatency

Syntax Gauge32

Max-Access	read-only
Status	current
Description	The round trip time (RTT) between the local and remote SAN Routers, measured in milliseconds, of the most recent connection keep-alive message. The RTT measurement is updated at one-third the interval of the rmtConnHoldTime of the connection.
Sequence	`::= { rmtConnectionEntry 13 }`

rmtConnLostConnections

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of initiator/target pair data connections, between the local and remote SAN Router, that had to be abnormally terminated. The reason the connections were terminated may be because the hold time expired, the port link status went down, or other action that caused lost connectivity on the link.
Sequence	`::= { rmtConnectionEntry 14 }`

rmtConnLostPackets

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of TCP segments sent by this SAN Router that had to be retransmitted.
Sequence	`::= { rmtConnectionEntry 15 }`

rmtConnTotalPackets

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of TCP segments transmitted by this SAN Router. This does not recount a segment that was retransmitted.

Sequence ::= { rmtConnectionEntry 16 }

rmtConnUpTime

Syntax TimeTicks

Max-Access read-only

Status current

Description The time, in hundredths of a second, since the current control connection to the remote SAN Router was established. This is the length of time that the current connection to the remote SAN Router has been up and running. If the remote SAN Router connection is currently down or disabled, this variable should return 0.

Sequence ::= { rmtConnectionEntry 17 }

rmtConnDiscoveredPathMtuSize

Syntax INTEGER (0 .. 65535)

Max-Access read-only

Status current

Description The Discovered Path Maximum Transmission Unit (MTU) Size, in bytes, is the MTU size that was discovered for the path used by this connection. The Path MTU is determined at connection initiation, and when the connection is made aware of route changes. The actual MTU used by the connnection is either the Discovered Path MTU or rmtConnMaxMtuSize, whichever is smaller. If the rmtConnMaxMtuSize has not been set (indicated by a value of 0), then it is ignored.

Sequence ::= { rmtConnectionEntry 18 }

rmtConnMaxMtuSize

Syntax INTEGER (0 .. 65535)

Max-Access read-only

Status current

Description The maximum value, in bytes, that can be used for the Maximum Transmission Unit (MTU) Size for the connection. This variable reflects what has been set for the TCP port that is hosting this

connection. The value for the port is set by using NISHAN-MGT fcswTCPMaxMtuSize. This variable can used to prevent accidental fragmentation when the Path MTU value, as returned by paths used by this connection, is incorrect causing accidental fragmentation. If this variable is set to 0, the default setting, then it is ignored.

Sequence `::= { rmtConnectionEntry 19 }`

rmtConnTcpWindowSize

Syntax INTEGER (0 .. 8184)

Max-Access read-create

Status current

Description DURABLE: { 0:all } The TCP window size used for all TCP data connections in this iFCP remote connection, in KBytes. The value configured here is rounded up to the next multiple of 8 KBytes internally.

The TCP window size specifies how much outstanding data (data not yet acknowledged by the receiver) may be sent on one TCP connection. The window size is also the receive buffer size for each TCP connection.

The recommended TCP window size is the expected bandwidth multiplied by the round-trip time (RTT). For example, the optimal window size for a 100 Mbit/sec connection with a 5 msec RTT is 100Mbit/sec * .005 sec = 0.5Mbits or about 65 KBytes.

The value 0 is a special value meaning 'Auto'. In Auto mode, the switch selects the optimal receive window size for each TCP connection independently. The window size is based on the TCP stack measured round-trip times. Each connection is adjusted dynamically as the connection's RTT changes. In the case of oversubscription, buffering resources are distributed fairly across connections to the extent possible. In the auto mode, each connection has a minimum window size of 128 KB and a maximum window size of 8184 KB.

The TCP window size may be manually changed at any time by settingthis MIB variable. The new value begins taking effect immediately. Large changes may take a few seconds to become fully effective, as the window size changes gradually. TCP window size changes do not interrupt the remote SAN Router traffic.

DEFVAL { 0 }

Sequence ::= { rmtConnectionEntry 20 }

rmtConnTcpTotalSlowStarts

Syntax Counter32

Max-Access read-only

Status current

Description The total number of TCP slow start congestion avoidance events (cumulative) incurred by the iFCP sessions (TCP connections) belonging to this remote SAN Router connection. The count is incremented for any iFCP login session (TCP connection) associated with this remote peer. This count is not changed when the iFCP sessions (TCP connections) come and go and simply reflects the condition between the pair of IP addresses in this remote peer connection. The remote peer connection is between the local SAN Router and a single remote SAN Router.

Sequence ::= { rmtConnectionEntry 21 }

rmtConnRowStatus

Syntax RowStatus

Max-Access read-create

Status current

Description This indicates the status of this entry. The row status is for creating the row entry. It does not show the status of the connection. The connection status is determined by the Action and Status attributes.

active (1),	read-write
notInService (2),	read-write
notReady (3),	read-only
createAndGo (4),	write-only
createAndWait (5),	write-only
destroy (6),	write-only

Sequence ::= { rmtConnectionEntry 22 }

rmtConnDuplicateSegments

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total number of duplicate TCP segments received on all TCP sessions belonging to this remote peer connection.
Sequence	`::= { rmtConnectionEntry 23 }`

rmtConnOutOfOrderSegments

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total number of out-of-order TCP segments received on all TCP sessions belonging to this remote peer connection.
Sequence	`::= { rmtConnectionEntry 24 }`

iFCP Backup Link Info

localPeer OBJECT IDENTIFIER ::= { nishanGtwyPeer 3 }

lclPeerRedundantSupportTable

Syntax	SEQUENCE OF LclPeerEntry
Max-Access	not-accessible
Status	current
Description	This table contains the list of backup iFCP connections learned from primary ports. Primary ports may be on this SAN Router or a different SAN Router. Backup ports on this SAN Router learn the iFCP connections from the primary ports, so the backup port is ready to take over in case the primary port goes away. These learned connections are not editable. The table is a fixed size, with the number of rows indicating the maximum number of backup iFCP connections supported. Unused

table entries have a value of '0.0.0.0' for lclPeerIpAddress and a value of 0 for lclPeerGigeIndex. Other variables are undefined for unused table entries.

Sequence `::= { localPeer 1 }`

lclPeerEntry

Syntax LclPeerEntry

Max-Access not-accessible

Status current

Description An entry in the table providing information about a local peer that this gatway is providing backup support for.

INDEX
{ lclPeerIndex } ::= { lclPeerRedundantSupportTable 1 }

LclPeerEntry ::= SEQUENCE {

lclPeerIndex	INTEGER,
lclPeerIPAddress	IpAddress,
lclPeerRemoteIPAddress	IpAddress,
lclPeerExpZoneListBitMap	OCTET STRING,
lclPeerConnectionStatus	INTEGER,
lclPeerGigeIndex	INTEGER,
lclPeerRemoteHoldTime	INTEGER,
lclPeerRmtSanID	Unsigned32,
lclPeerRcvBytes	Counter32,
lclPeerXmtBytes	Counter32,
lclPeerConnections	Gauge32,
lclPeerLatency	Gauge32,
lclPeerLostConnections	Counter32,
lclPeerLostPackets	Counter32,
lclPeerTotalPackets	Counter32,
lclPeerUpTime	TimeTicks,

lclPeerFailoverCount	Unsigned32,
lclPeerDiscoveredPathMtuSize	INTEGER,
lclPeerTcpWindowSize	INTEGER}

lclPeerIndex

Syntax	INTEGER(0..2147483647)
Max-Access	read-only
Status	current
Description	An arbitrary integer to number backup links.
Sequence	`::= { lclPeerEntry 1 }`

lclPeerIPAddress

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The inband IP Address of the primary port that provided this backup link information. If this table entry is unused, lclPeerIPAddress is '0.0.0.0'.
Sequence	`::= { lclPeerEntry 2 }`

lclPeerRemoteIPAddress

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The IP Address of the iFCP port that the local port is exporting zones to.
Sequence	`::= { lclPeerEntry 3 }`

lclPeerExpZoneListBitMap

Syntax	OCTET STRING (SIZE (64))
Max-Access	read-only

Status	current
Description	A bit map indicating the zones exported from the local peer SAN Router to the remote peer SAN Router indicated. These are the zones which this SAN Router would need to export if the local peer SAN Router failed. Zone 1 is represented by the least significant (rightmost) bit of byte 0.
Sequence	`::= { lclPeerEntry 4 }`

lclPeerConnectionStatus

Syntax	INTEGER { down(0), idle(1), primaryactive(2), failoveractive(3) }
Max-Access	read-only
Status	current
Description	This indicates the current status of the connection. down(0) - backup connection is not operating. idle(1) - backup connection is ready but not activated. primaryactive(2) - backup connection is ready but not activated. failoveractive(3) - backup connection is active and running.The precise differences between values 1 and 2 are not known.
Sequence	`::= { lclPeerEntry 5 }`

lclPeerGigeIndex

Syntax	INTEGER (1 .. 5)
Max-Access	read-only
Status	current
Description	The Gigabit Ethernet port index used for this backup connection. If this table entry is unused, the value is 0.
Sequence	`::= { lclPeerEntry 6 }`

lclPeerRemoteHoldTime

Syntax	INTEGER (10 .. 90)
Max-Access	read-only
Status	current
Description	The hold time, in seconds, used between the local and remote peers.

Sequence ::= { lclPeerEntry 7 }

lclPeerRmtSanID

Syntax Unsigned32

Max-Access read-only

Status current

Description The SAN ID of the remote SAN. The format is a 4-Byte unsigned value. This is determined throught inter-SAN Router communication.

Sequence ::= { lclPeerEntry 8 }

lclPeerRcvBytes

Syntax Counter32

Max-Access read-only

Status current

Description The total number of bytes received from this remote SAN Router since the remote SAN Router's connection was first established, including MAC headers and CRC. For some devices this count may be only approximate, since the source SAN Router is only identified after the MAC layer fields have been stripped off.

Sequence ::= { lclPeerEntry 9}

lclPeerXmtBytes

Syntax Counter32

Max-Access read-only

Status current

Description The total number of bytes transmitted by this remote SAN Router since the remote SAN Router's connection was first established, including MAC headers and CRC. For some devices this count may be only approximate, since the destination SAN Router is determined before the MAC layer fields are added.

Sequence ::= { lclPeerEntry 10}

lclPeerConnections

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The current number of initiator/target pair data connections, between the local and remote SAN Router, using the iFCP link. For example, if one local server uses four remote disks, four connections are counted.
Sequence	`::= { lclPeerEntry 11}`

lclPeerLatency

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The round trip time (RTT) between the local and remote SAN Routers, measured in milliseconds, of the most recent connection keep-alive message. The RTT measurement is updated at one-third the interval of the rmtConnHoldTime of the connection.
Sequence	`::= { lclPeerEntry 12}`

lclPeerLostConnections

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of times that the control connection to this remote SAN Router had to be abnormally terminated because the hold time expired, the port link status went down, or other indications of lost connectivity. This is a cumulative total of fatal network failures.
Sequence	`::= { lclPeerEntry 13}`

lclPeerLostPackets

Syntax	Counter32

Max-Access read-only

Status current

Description The number of TCP segments sent by this SAN Router that had to be retransmitted.

Sequence `::= { lclPeerEntry 14}`

---

lclPeerTotalPackets

Syntax Counter32

Max-Access read-only

Status current

Description The number of TCP segments transmitted by this SAN Router. This does not recount a segment that was retransmitted.

Sequence `::= { lclPeerEntry 15}`

---

lclPeerUpTime

Syntax TimeTicks

Max-Access read-only

Status current

Description The time, in hundredths of a second, since the current control connection to the remote SAN Router was established. This is the length of time that the current connection to the remote SAN Router has been up and running. If the remote SAN Router connection is currently down or inactive, this variable should return 0.

Sequence `::= { lclPeerEntry 16}`

---

lclPeerFailoverCount

Syntax Unsigned32

Max-Access read-only

Status current

Description The number of failover events to this connection that have occurred since the connection to the remote SAN Router was established (lclPeerUpTime). If the link to the remote SAN Router goes down and is then restablished, then the failover count is reset to 0.

Sequence ::= { lclPeerEntry 17}

lclPeerDiscoveredPathMtuSize

Syntax INTEGER (0 .. 65535)

Max-Access read-only

Status current

Description The Discovered Path Maximum Transmission Unit (MTU) Size, in bytes, is the MTU size that was discovered for the path used by this connection. The Path MTU is determined at connection initiation, and when the connection is made aware of route changes. The actual MTU used by the connnection is either the Discovered Path MTU or rmtConnMaxMtuSize, whichever is smaller. If the rmtConnMaxMtuSize has not been set (indicated by a value of 0), then it is ignored.

Sequence ::= { lclPeerEntry 18}

lclPeerTcpWindowSize

Syntax INTEGER (0 .. 8184)

Max-Access read-only

Status current

Description The TCP window size used for all TCP data connections in this iFCP remote connection, in KBytes. The TCP window size specifies how much outstanding data (data not yet acknowledged by the receiver) may be sent on one TCP connection. The window size is also the receive buffer size for each TCP connection.

The value 0 is a special value meaning 'Auto'. In Auto mode, the switch selects the optimal receive window size for each TCP connection independently. The window size is based on the TCP stack measured round-trip times. Each connection is adjusted dynamically as the connection's RTT changes. In the case of oversubscription, buffering resources are distributed fairly across connections to the extent possible. In the automode, each connection has a minimum window size of 128 KB and a maximum window size of 8184 KB.

Sequence ::= { lclPeerEntry 19 }

END

D

McDATA iSCSI Configuration MIB

McDATA iSCSI Configuration MIB

This MIB contains management objects for McDATA iSCSI configuration.

Version 1.1

Revision history:

5/29/01 Initial version released.

6/06/01 Removed iscsiSessionTable. It duplicates info in the standard iSCSI MIB.

6/20/01 Moved origin from {nishanMgmt 13} to {nishan 10}

6/26/01 Changed size for iscsiDevName and iscsiDevAlias

6/28/01 Made iscsiDevAlias read-create instead of read-only, added iscsiDevRole, added isnsLocateTable, added isnsAddressTable, added iscsiZone2DDTable, increased MIB version to 1.1

06/29/01 Added "both" enumerated value to iscsiDevRole

07/09/01 Updates from MIB review- move typedefs to NISHAN-SMI, add more default values, reword some descriptions.

07/11/01 FcWWN and FcID typedefs were renamed as WWNtype and FCIDtype for backward compatibility.

07/24/01 Removed isnsLocateTable (duplicated in IETF iSNS MIB) remove port index from isnsAddressTable (no longer port-specific) change iscsiZone2DDTable index from arbitrary integer to zone ID.

07/31/02 Added isciDevAdminStatus and iscsiInitiatorAutoRegister to auto-configure iSCSI initiators.

08/07/02 Removed iSNS portion of MIB (duplicate of iSNS MIB) removed imports referenced in the iSNS portion of the mib.

03/12/03 Included support for IPS 5000 series.

11/06/03 Added iscsiDevicesMaxEntries and iscsiDevicesCurrentEntries

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

```
NISHAN-ISCSI DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE, IpAddress
        FROM SNMPv2-SMI
DisplayString, RowStatus
        FROM SNMPv2-TC
nishan, WWNtype, FCIDtype
        FROM NISHAN-SMI;
nishanISCSI     MODULE-IDENTITY
Last-Updated    0509190000Z
Organization    McDATA Corporation
Contact-Info    McDATA Corporation
                380 Interlocken Crescent
                Broomfield, CO 80021 USA
                Tel : 1 720 558-8000
                Fax : 1 720 558-3860
                email : information@mcdata.com
```

Description	The private MIB for Nishan iSCSI device configuration.
Revision	0509190000Z
Description	Released for E/OSi version 4.7.
Revision	0507270000Z
Description	Update descriptions and comments to remove references to old IPS product lines.
Revision	0105220000Z
Description	Initial public release of this MIB module.
Sequence	`::= { nishan 10 }`

iSCSI Device Configuration

iSCSI Device Table

This table contains one row for each iSCSI device connected to this SAN Router. The table shows the mapping from the iSCSI description to the mFCP information registered in SNS. iSCSI devices are not discovered. They must be manually configured on the switch, so the SAN Router can register with SNS as a proxy for the iSCSI devices.

The user must manually configure the iSCSI Name, IP address, IP port, and switch port in the table below. The iSCSI Alias is learned dynamically from the device, but may be supplied by the user if not configured on the device. The port and node WWNs and SoIP socket number are generated by the SAN Router.

iscsiDeviceTable

Syntax	SEQUENCE OF IscsiDeviceEntry
Max-Access	not-accessible
Status	current
Description	This table contains information on how each iSCSI device connected to this SAN Router is mapped to SNS registration information.
Sequence	`::= { nishanISCSI 1 }`

iscsiDeviceEntry

Syntax	IscsiDeviceEntry

Max-Access	not-accessible
Status	current
Description	Entry containing info for a particular iSCSI device.
	INDEX { iscsiDevIndex }
Sequence	`::= { iscsiDeviceTable 1 }`
	IscsiDeviceEntry ::= SEQUENCE {
	iscsiDevIndex INTEGER,
	iscsiDevName DisplayString,
	iscsiDevIpAddress IpAddress,
	iscsiDevTcpPort INTEGER,
	iscsiDevSwitchPort INTEGER,
	iscsiDevAlias DisplayString,
	iscsiDevRole INTEGER,
	iscsiDevPortWWN WWNtype,
	iscsiDevNodeWWN WWNtype,
	iscsiDevSoIPSocket FCIDtype,
	iscsiDevAdminStatus INTEGER,
	iscsiDevRowStatus RowStatus }

iscsiDevIndex

Syntax	INTEGER (1..500)
Max-Access	read-only
Status	current
Description	An arbitrary index value that identifies this iSCSI device.
Sequence	`::= { iscsiDeviceEntry 1 }`

iscsiDevName

Syntax	DisplayString (SIZE (1..255))
Max-Access	read-create
Status	current

Description DURABLE: { 0:all }

The unique iSCSI device name, or the default name 'iscsi'. iSCSI names must be unique within the SAN. iSCSI names are intended to be world-wide unique. True names may be either fully qualified names (e.g., iqn.com.acme.sn333) or an IEEE EUI name (e.g., eui.02004567A25678D). See internet draft draft-ietf-ips-iscsi-name-disc-02.txt for more details of iSCSI names.

When the iSCSI device is registered in SNS, the iSCSI name is used for the Symbolic Node Name.

Sequence `::= {iscsiDeviceEntry 2}`

---

iscsiDevIpAddress

Syntax IpAddress

Max-Access read-create

Status current

Description DURABLE: { 0:all }

The IP address of the iSCSI device.

Sequence `::= {iscsiDeviceEntry 3}`

---

iscsiDevTcpPort

Syntax INTEGER (0..65535)

Max-Access read-create

Status current

Description DURABLE: { 0:all }

The IP port number for the iSCSI device. IANA will assign a default port number for iSCSI devices, but for now different manufacturers use different ports. The port may be 0 for initiators.

Sequence `::= {iscsiDeviceEntry 4}`

---

iscsiDevSwitchPort

Syntax INTEGER (1..16)

Max-Access read-create

Status	current
Description	DURABLE: { 0:all }
	The local SAN Router's port to be used for traffic to the iSCSI device, if the iSCSI device is not on a locally-attached subnet. The SAN Routers perform only static routing (they do not learn routes on iSCSI interfaces by listening to route advertisements). The default gateway specified for the port identified by this variable becomes the next hop for outgoing iSCSI traffic.
Sequence	`::= {iscsiDeviceEntry 5}`

iscsiDevAlias

Syntax	DisplayString (SIZE (0..80))
Max-Access	read-create
Status	current
Description	DURABLE: { 0:all }
	A user-friendly name to describe this iSCSI device. The alias may be set manually on the SAN Router, or configured on the iSCSI device and learned by the SAN Router. If an alias is configured on the iSCSI device, it will replace an alias entered on the SAN Router. The alias becomes the port symbolic name when the device is registered with SNS.
	DEFVAL { "" }
Sequence	`::= {iscsiDeviceEntry 6}`

iscsiDevRole

Syntax	INTEGER { initiator(1), target(2), both(3) }
Max-Access	read-create
Status	current
Description	DURABLE: { 2:all }
	Specifies whether the iSCSI device is an initiator, a target, or both. The 'both' option allows a device that shares the same IP address for target and initiator functions to be entered in the iscsiDeviceTable just once. Devices that use different IP addresses for the initiator and

target functions, or initiators that use more than one IP address, are entered in the iscsiDeviceTable once for each address.

DEFVAL { target }

Sequence `::= {iscsiDeviceEntry 7}`

---

iscsiDevPortWWN

Syntax WWNtype

Max-Access read-only

Status current

Description The port World-Wide Name assigned to this iSCSI device. The SAN Router generates this WWN in order to register the iSCSI device with the Storage Name Service.

Sequence `::= {iscsiDeviceEntry 8}`

---

iscsiDevNodeWWN

Syntax WWNtype

Max-Access read-only

Status current

Description The node World-Wide Name assigned to this iSCSI device. The SAN Router generates this WWN in order to register the iSCSI device with the Storage Name Service.

Sequence `::= {iscsiDeviceEntry 9}`

---

iscsiDevSoIPSocket

Syntax FCIDtype

Max-Access read-only

Status current

Description The IP Storage socket number generated by the SAN Router to register the iSCSI device with the Storage Name Service. The socket number for FC devices is the 3-byte FCID.

Sequence `::= {iscsiDeviceEntry 10}`

iscsiDevAdminStatus

Syntax	INTEGER {enable(1), disable(2)}
Max-Access	read-create
Status	current
Description	DURABLE: {enable} This field is used to enable/disable the admin status of the iSCSI device. If enabled, then the device is registered into SNS.
Sequence	`::= { iscsiDeviceEntry 11 }`

iscsiDevRowStatus

Syntax	RowStatus
Max-Access	read-create
Status	current
Description	This variable is used to create and delete rows in iscsiDeviceTable, according to the SNMPv2 RowStatus convention. Possible values are:

active (1),	read-write
notInService (2),	read-write
notReady (3),	read-only
createAndGo (4),	write-only
createAndWait (5),	write-only
destroy (6),	write-only

To delete an iSCSI device, set iscsiDevRowStatus to destroy(6). To create a new iSCSI device, the agent requires this sequence:

1. set iscsiDevRowStatus to createAndWait(5) for an unused index value.
2. set other column values.
3. set iscsiDevRowStatus to active(1).

The values notInService(2), notReady(3), and createAndGo(4) are not supported. If any column, including iscsiDevName or iscsiDevIpAddress, is updated by setting the variable with an existing index, then the existing SNS information (WWNs and SoIP

socket) are used for the new iSCSI information. If a table row is deleted and re-added, new SNS information (WWNs and SoIP socket) may be created.

Sequence `::= {iscsiDeviceEntry 12}`

---

iscsiInitiatorAutoRegister

Syntax INTEGER {true(1), false(2)}

Max-Access read-write

Status current

Description DURABLE: {false}

This scalar is used to determine if all initiator logins should automatically be allowed to register with SNS.

Sequence `::= { nishanISCSI 2 }`

---

iscsiDevicesMaxEntries

Syntax INTEGER (1..512)

Max-Access read-only

Status current

Description This scalar is used to report the maximum number of iSCSI devices supported in the switch. In the v4.x release, the maximum number of iSCSI devices is 50.

Sequence `::= { nishanISCSI 3 }`

---

iscsiDevicesCurrentEntries

Syntax INTEGER (0..512)

Max-Access read-only

Status current

Description This scalar is used to report the number of iSCSI devices that are currently configured, manually or automatically logged, in the switch.

Sequence `::= { nishanISCSI 4 }`

END

E

McDATA Eclipse SAN Router Management MIB

NISHAN-MGT.MIB

This MIB contains management objects for McDATA Eclipse SAN Routers, formerly Nishan products. See the MODULE-IDENTITY macro for the current revision of this file.

Version 1.0

Revision history:

4/20/01:	P1 and Titan objects added so this MIB can be used to manage all Nishan products.
4/25/01:	Added comments to indicate fcswALPortMappingTable and fcswSNSUnicastCfgTable are not yet implemented.
5/17/01:	Added comments to the Card Table.PartNumber description.
5/22/01:	Added SNS version info (fcswSNSrevInfo), Zoneset info (fcswSNSzsInfo), and TCP Port support (fcswTCPPortTable, fcswPortSupportsTCP, new port types).
6/2/01:	Added DURABLE clause for above objects to be saved to falsh.
6/7/01:	Changed rnage of fcswSNSzstTable to 1 to 64.
6/12/01:	Added logMsgUsedSpace.
6/13/01:	Added range to logMsgUsedSpace to fix smicng warning.

7/16/01: Added new enumerated values to fcswPortType.

8/14/01: Added flashImageOnNextReset.

9/14/01: Added omniPort value to fcswNsPortType and elaborated its definition.

9/27/01: Added fcswTCPAutoReset and fcswTCPResetNow.

11/14/01: Added fcswTCPOptions.

01/28/02: Added fcswEportSwitchInterconnect (implemented since 9/25/01).

02/21/02: Added updated fcswTCPOptions with compression levels and flow-control.

03/01/02: Added lport (private loop port) to the port type tables.

03/07/02: Added fcswTCPCompressionStatsTable for the TCP Port compression stats.

03/13/02: Changed description of the compression types in fcswTCPOptions.

03/15/02: Updated fcswTCPOptions to enable jumboframes bit.

03/19/02: Added systemOptions.

04/12/02: Updated fcswTCPCompressionStatsTable entries descriptions.

04/19/02: Added nishanfcswSnsLuns for LUN Mapping/Masking.

04/26/02: Added fcswSNSLunMapping variable to enable/disable LUN Mapping/Masking.

05/07/02: Corrected SMICNG warnings and errors.

05/13/02: Added Eport AggregateId.

06/14/02: Added fcswTCPMaxMtuSize to configure the MTU size per TCP port.

06/25/02: Moved variables/tables under nishanfcswSnsLuns to nishanFCsw branch.

06/26/02: Updated fcswTCPOptions to support MTU size configurations.

07/01/02: Added fcswLunNumber to fcswLunInfoTable table to denote the 2 byte physical LUN number. changed fcswPLunNumber to fcswPLunIndex in the fcswLunMappingTable to indicate the sequential lun index.

07/30/02: Fixed case error in fcswEportAllowFcPortZoning.

09/04/02: Updated fcswTCPOptions description for AutoFastWrite bit

10/04/02: Updated comment strings for LUNInfo table to have "=" instead of "-". To compile our mibs into HP NNM correctly, we recommend using "=" in the comment strings. Apparently, if you have "-", in the comments, then they should be even in number, so NNM can treat a pair of "-" as comment.

10/28/02: Added fcswEPortZoneSetActionStatus to return the status codes for the e-port zone set action. Added fcswEPortSoIPZoneCleanup to remove ny appended zones. Changed fcswEPortMakeSoIPActive mib variable name to fcswEPortSoIPZoneSetPolicy so that the name is consistent with the action it does. Added fcswEPortNeighborEDTOV and fcswEPortNeighborRATOV variables to get the connected FC neighbor's timeout values.

10/29/02: Added fcswZoneBurstLimit and fcswZoneNormalLimit for bandwidth manager support. Added fcswZoneType for iSNS support.

11/20/02: fcswEPortZoneSetActionStatus variable . Added description to delete or invalidate a LUN MAP. Changed Burst Limit and Normal limit ranges.

11/26/02: Added fcswTCPStorageStatsTable to show storage statistics for the TCP ports. Added fcswTCPSessionsTable to list all the sessions for a TCP port. Added fcswTCPSessionStatsTable to show storage statistics for the TCP sessions.

12/05/02: Added fcswNsFc4Features in the fcswNsPortTable to capture the FC4 feature bits as defined in FC-GS-3.

12/23/02: Added fcswTCPiSCSIOptions, fcswTCPiSCSIFirstBurstLength, fcswTCPiSCSIMaxBurstLength, fcswTCPiSCSIMaxRecvDataSegmentLength variables to configure iSCSI options per TCP port

1/18/03:	Add fcswPortActualSpeed to determine the actual port speed when the port speed is configured to auto. Added mgtPortReset mib variable to reset the mgmt port after changing the IP address. Aded comments to explain which of the variables are reserved for future use.
1/20/03:	Added fast ethernet enumerations to the fcswPortType; add twoxutpgbic enumeration to the fcswCardType.
1/22/03:	Changed fcswPortConfiguredSpeed range Syntax to be acceptable to SMICNG.
1/23/03:	Reserve fcswPortType enumerations for later use.
1/27/03:	Changed TCP Port compression statistic variable descriptions to reflect units change. Counters now return units of 16 bytes instead of 128K bytes.
02/11/03:	Changed comment lines to support HP NNM. i.e. replace "-" with "=".
02/28/03:	Add connector type HSSDC2 to fcswPortConnectorType
03/03/03:	Expand fcswPortConnectorType and fcswPortType description
03/12/03:	add fcswTcpOptions bit to enable SACK for iSCSI. Update fcswTcpOptions description for compression bits for clarity. Add new variable fcswTCPCompressionMethod to select compression method.
03/12/03:	Included support for IPS 5000 series and 3350 models. Updated fcswTcpOptions bitmap to support iSNS changes. Added a new variable fcswTCPiSNSServerIpAddress to fcswTCPPortTable to set the iSNS server's IP address if the port is configured as iSNS client.
03/17/03:	Added fcswTCPiSCSITargetChapCredential to fcswTCPPortTable to configure the CHAP authentication secret key for virtual targets.
04/11/03:	Defined two more bits in fcswTCPiSCSIOptions to select the iSCSI authentication method.
04/18/03:	Corrected the Syntax range for fcswPortUtilization.

04/21/03: Added another compression method to fcswTCPCompressionMethod.

04/28/03: Modified the fcswEPortSoIPZoneCleanup variable's allowed values and its description

05/09/03: Modified the fcswChasFanOperStatus enumerations and descriptions

05/21/03: Add fcswTcpOptions bit to enable SACK for iFCP.

05/21/03: Update SNS configuration variable descriptions

05/29/03: Obsolete fcswChasFanPartNumber

06/27/03: Added fIscsiGtwy to fcswNsPortType in the SNS table. This type indicates the iSCSI listener for each of the configured iSCSI ports.

07/08/03: Added fcswEPortErrorMsgString to fcEPortTable. This octet string allows a more detailed error message to describe an error encountered on a given Eport.

11/19/03: Added softwareBundleString to show the software string associated with the version of software running on the switch.

03/25/04: Updated fcswTCPResetNow description to specify a return value to indicate when the port reset is finished.

04/05/04: Changed fabricManager to fabric controller, and placed descriptions throughout the mib to describe when a port must be taken offline for a configuration to take place.

07/14/04: Added fcswEportFcNodeWWN to report McDATA OUI support for E/OS 5.0.

07/15/04: Added fcswTCPiSCSILoginRetryTimeout to control the number of retries an iSCSI Initiator attempts to relogin to the target.

07/19/04: Fixed description of fcswPortOperStatus (bug #4684)

09/27/04: Added inTransition(5) value to fcswPortOperStatus (MCDT-00027533).

11/16/04: Added fcswSnsClientTable.

01/25/05 Add 4 new bit definitions to the fcswTCPOptions bitmap, and one new variable (fcswTCPReorderThreshold) to fcswTcpPortTable, to implement "Storage-Friendly TCP".

03/10/05 Add another new bit definition to the fcswTCPOptions bitmap, for the “Reduced Slow Start Timeout" feature.

03/29/05 Add a bit to systemOptions bitmap to disable Telnet.

04/08/05 Removed ancient un-implemented variables.

04/18/05 Add loginBannerPart1 and loginBannerPart2

04/20/05 Change banner entry to a table.

04/21/05 Change banner table to be a table of lines.

04/29/05 Change banner table usage.

After 05/05/05, changes are recorded by adding another Revision clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

```
NISHAN-MGT DEFINITIONS ::= BEGIN
IMPORTS
MODULE-IDENTITY,
OBJECT-TYPE,
        IpAddress,
        Integer32,
        Gauge32,
        Counter32
FROM SNMPv2-SMI
        DisplayString,
        DateAndTime,
        RowStatus,
        PhysAddress,
        MacAddress,
        TruthValue,
TEXTUAL-CONVENTION
        FROM SNMPv2-TC
nishan
        FROM NISHAN-SMI
```

```
                InetAddressType,
                InetAddress
            FROM INET-ADDRESS-MIB
            ;nishanMgmt     MODULE-IDENTITY
Last Updated    0509290000Z
Organization    McDATA Corporation
Contact info    McDATA Corporation
                380 Interlocken Crescent
                Broomfield, CO 80021 USA
                Tel : 1 720 558-8000
                Fax : 1 720 558-3860
                email : information@mcdata.com
Description     The private MIB for McDATA Eclipse SAN Router Management.
                Revisions are listed in reverse chronological order.
Revision        0509290000Z
Description     Updated the descriptions of mib objects
                fcswTCPXmitBufferMgmtMem,
                fcswTCPXmitMgmtMaxBufferCount, and
                fcswTCPXmitMgmtCurrentUsedBufferCount.
Revision        0509190000Z
Description     Released for E/OSi version 4.7.
Revision        0509060000Z
Description     1. Added mib object fcswTCPXmitMgmtMaxBufferCount and
                fcswTCPXmitMgmtCurrentUsedBufferCount.
Revision        0508190000Z
Description     1. Added a new bit definition to fcswTCPOptions to enable or disable
                Transmit Buffer Management.

                2. Added new fcswTCPXmitBufferMgmtMem to configure the
                number of buffers for Transmit Buffer Management.
Revision        0507270000Z
Description     Update description of fcswPortAutoLinkAggregationAdminStatus to
                indicate that link aggregation is not supported in E/OSi 4.7.
```

Revision 0507180000Z

Description Improve descriptions for fcswTCPiSCSIFirstBurstLength, fcswTCPiSCSIMaxBurstLength, fcswTCPiSCSIMaxRecvDataSegmentLength, fcswZoneBurstLimit, and fcswZoneNormalLimit.

Revision 0506280000Z

Description Added fcswTCPFastWriteOptions, fcswTCPFastWriteMaxXferRdys, and fcswTCPFastWriteBufferMem to configure FastWrite properties.

Revision 0506150000Z

Description Multiple corrections to various descriptions, from PTI. Removed more unimplemented variables and tables.

Revision 0506090000Z

Description Multiple corrections to various descriptions. Removed unimplemented variables and tables. Increased minimum value for fcswZoneBurstLimit and fcswZoneNormalLimit.

Revision 0506060000Z

Description Corrections to 'beacon' description from PTI.

Revision 0505090000Z

Description Increased size range on flashImageVersion.

Revision 200101200000Z

Description Initial public release of this MIB module

Sequence `::= { nishan 1 }`

FCIDtype ::= OCTET STRING (SIZE (3))

WWNtype ::= OCTET STRING (SIZE (8))

NishanOperStatus

::= TEXTUAL-CONVENTION

Status current

Description Represents the state of device being monitored

Syntax INTEGER {

disabled(0),

normal(1),

warning(2),
critical(3),
shutdown(4),
notpresent(5) }

Port configuration

Except for specific variables marked below, this section is supported on all Nishan products.

nishanPort OBJECT IDENTIFIER ::= { nishanMgmt 2 }

portSetUp OBJECT IDENTIFIER ::= { nishanPort 1 }

Management Port configuration

mgtPort OBJECT IDENTIFIER ::= {portSetUp 1}

mgtPortIpAddressOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	The IP address to be used for this management port after the next reset.
Sequence	`::= {mgtPort 1}`

mgtPortNetMaskOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	The network mask to be used for the management port after the next reset.
Sequence	`::= {mgtPort 2}`

mgtBeacon

Syntax	INTEGER { on(1), off(2) }
Max-Access	read-write
Status	current
Description	Set to on (1) to cause the management port status LED to blink on the hardware front panel. This may be used to locate a particular SAN Router in a rack of equipment. This feature is only supported on the Eclipse Model 3300. DEFVAL {off}
Sequence	`::= {mgtPort 3}`

mgtPortIpAddressCurrent

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The IP address in use for this management port.
Sequence	`::= {mgtPort 4}`

mgtPortNetMaskCurrent

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The network mask in use for this management port.
Sequence	`::= {mgtPort 5}`

mgtPortOperStatus

Syntax	INTEGER { up(1), down(2) }
Max-Access	read-write
Status	current

Description The operational status of the mgmt port. This tells you if the interface is up or down.

Sequence `::= {mgtPort 6}`

mgtPortPhysAddress

Syntax PhysAddress (SIZE (6))

Max-Access read-only

Status current

Description The mgmt port MAC address.

Sequence `::= {mgtPort 7}`

mgtPortIPAddressViaDHCP

Syntax INTEGER { false(0), true (1) }

Max-Access read-write

Status current

Description Set true(1) to obtain the management port address via DHCP. This feature is not currently supported.

Sequence `::= {mgtPort 8}`

mgtPortBroadcastAddressOnNextReset

Syntax IpAddress

Max-Access read-write

Status current

Description The Broadcast address in use for this management port after the next reset. This variable does not need to be used, for 2 reasons. (1) The broadcast address defaults to a subnet broadcast address as determined by the port address and subnet mask. (2) The supported management protocols do not issue broadcasts.

Sequence `::= {mgtPort 9}`

mgtPortBroadcastAddressCurrent

Syntax IpAddress

Max-Access	read-only
Status	current
Description	The Broadcast address in use for this management port.
Sequence	`::= {mgtPort 10}`

mgtPortDefaultGatewayAddressOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current
Description	The new next-hop address for a permanent static route. This value remains even if the SAN Router is reset to factory defaults. This variable, together with nearby variables, provides an IP static route to the primary management station so that contact is not lost even when the SAN Router is reset to factory default. The value set here takes effect the next time the SAN Router is reset.
Sequence	`::= {mgtPort 11}`

mgtPortCurrentDefaultGatewayAddress

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The current next-hop address for a permanent static route. This value remains even if the SAN Router is reset to factory defaults. This variable, together with nearby variables, provides an IP static route to the primary management station so that contact is not lost even when the SAN Router is reset to factory default.
Sequence	`::= {mgtPort 12}`

mgtPortHostIpAddressOnNextReset

Syntax	IpAddress
Max-Access	read-write
Status	current

Description The IP address of NMS in use to management this management port to which this entry is applicable after the next reset.

Sequence `::= {mgtPort 14}`

---

mgtPortCurrentHostIpAddress

Syntax IpAddress

Max-Access read-only

Status current

Description The current destination address of a permanent static route. This value remains even if the SAN Router is reset to factory defaults. This variable, together with nearby variables, provides an IP static route to the primary management station so that contact is not lost even when the SAN Router is reset to factory default.

Sequence `::= {mgtPort 15}`

---

mgtPortHostNetMaskOnNextReset

Syntax IpAddress

Max-Access read-write

Status current

Description The new subnet mask for a permanent static route. This value remains even if the SAN Router is reset to factory defaults. This variable, together with nearby variables, provides an IP static route to the primary management station so that contact is not lost even when the SAN Router is reset to factory default. The value set here takes effect the next time the SAN Router is reset.

Sequence `::= {mgtPort 16}`

---

mgtPortCurrentHostNetMask

Syntax IpAddress

Max-Access read-only

Status current

Description The current subnet mask for a permanent static route. This value remains even if the SAN Router is reset to factory defaults. This variable, together with nearby variables, provides an IP static route to

the primary management station so that contact is not lost even when the SAN Router is reset to factory default.

Sequence `::= {mgtPort 17}`

mgtPortReset

Syntax INTEGER {false(0), true(1)}

Max-Access read-write

Status current

Description Reset the management port. This variable is not supported on current Eclipse models.

DEFVAL {false}

Sequence `::= {mgtPort 18}`

System configuration

nishanSystem OBJECT IDENTIFIER ::= { nishanMgmt 3 }

memoryUtil

Syntax INTEGER (0 .. 100)

Max-Access read-only

Status current

Description Shared Memory utilization of the device.

Sequence `::= { nishanSystem 1 }`

beacon

Syntax INTEGER { false(0), true(1) }

Max-Access read-write

Status current

Description Set to true(1) to blink the system LEDs on the Eclipse IPS 3300. Not supported on other Eclipse models

DEFVAL {false}

Sequence ::= { nishanSystem 2 }

timeSetup

Syntax DateAndTime

Max-Access read-write

Status current

Description Use this value to set up the system date/time.

Sequence ::= { nishanSystem 4 }

readOnlyPassword

Syntax DisplayString

Max-Access read-write

Status current

Description Read community string. As the value is set, it takes effect immediately -- not after the next reset.

DEFVAL {"public"}

Sequence ::= { nishanSystem 5 }

systemReset

Syntax INTEGER { false(0), true(1) }

Max-Access read-write

Status current

Description Setting this variable to true will reset the system.

DEFVAL {false}

Sequence ::= { nishanSystem 9 }

systemNsPrimaryStatus

Syntax INTEGER { snsunvailable(1), snsavailable(2) }

Max-Access read-only

Status current

Description	SAN Router Name Server is available or not.
	DEFVAL {snsavailable}
Sequence	`::=  { nishanSystem 11 }`

systemSaveToFlash

Syntax	INTEGER { yes(1), no(2) }
Max-Access	read-write
Status	current
Description	Setting this variable to 1 causes the System configuration to be saved to flash.
	DEFVAL {no}
Sequence	`::= { nishanSystem 12 }`

flashNvramVersion

Syntax	INTEGER (0..255)
Max-Access	read-write
Status	current
Description	SequenceDURABLE: NVRAM version number read from flash.
Sequence	`::= { nishanSystem 13 }`

currentNvramVersion

Syntax	INTEGER (0..255)
Max-Access	read-only
Status	current
Description	current NVRAM version that should be used.
Sequence	`::= { nishanSystem 14 }`

configurationStatus

Syntax	INTEGER { touched(1), saved(2), noChanges(3) }
Max-Access	read-only

Status current

Description Status of system configuration data:

touched(1): configuration has been modified but not yet saved to flash.

saved(2): all configuration changes have been saved.

noChanges(3): configuration has not been changes since last reset.

DEFVAL {noChanges}

Sequence `::= { nishanSystem 15 }`

systemLeds

Syntax DisplayString

Max-Access read-only

Status current

Description A character string that contains the values of all the system LEDs. Each character represents an LED. If the LED values are not available, a zero length string is returned. The mapping of character position to front-panel LED position depends on the specific Eclipse model. Each character may be:

value	meaning
0	off
1	slow blinking
2	fast blinking
3	on

Sequence `::= { nishanSystem 16 }`

systemOptions

Syntax INTEGER (0..2147483647)

Max-Access read-write

Status current

Description DURABLE: { 0:all }

A bitmap representing options for this SAN Router. Bit 0 is the least significant bit. Setting a bit enables the option. Current bit assignments are:

bit 0:	Enable Jumbo frames on local GE ports. (This is not available in E/OSi 4.7 release.)
bit 1:	Enable R_Port SAN IDs. (Not used in E/OSi 4.4 and later
bits 2-31:	Reserved
bit 2	Internal-use
bit 3	Telnet turning the bit on disables the feature and vice versa
bit 4	Reserved
bit 5	Internal-use
bit 6	Internal-use
bit 7	Internal-use
bits 8-31	Reserved

Reserved bits must be zero when read, and must be ignored by the agent when set, for future backward compatibility.

DEFVAL { 0 }

Sequence `::= {nishanSystem 17}`

---

softwareBundleString

Syntax DisplayString

Max-Access read-only

Status current

Description This value will reflect the supported features for the version of the software running on this switch. For example: if the s/w version is 4.2.SI then this string will have the value 'Standard iSCSI'

DEFVAL {"Standard iSCSI"}

Sequence `::= {nishanSystem 19}`

---

loginBannerLines

Syntax Integer32 (0..25)

Max-Access read-only

Status current

Description The current number of lines in the login banner. This is also the number of entries in the loginBannerTable.

Sequence `::= {nishanSystem 20}`

loginBannerTable

Syntax SEQUENCE OF LoginBannerEntry

Max-Access not-accessible

Status current

Description The loginBannerTable defines a text banner to display before the login prompt.

Each table entry is a text string representing one line of the banner. Blank lines must contain at least one space character. To set the prompt, write each line starting at index 1. If the prompt is less than 25 lines long, terminate the prompt by writing a zero-length string after the last prompt line. The zero-length string deletes all following lines. The zero-length string is not returned when reading the table. The zero-length string is not counted in the prompt line count in variable loginBannerLines.

Sequence `::= { nishanSystem 21 }`

loginBannerEntry

Syntax LoginBannerEntry

Max-Access not-accessible

Status current

Description One line of the login banner.

INDEX {loginBannerLineNumber}

Sequence `::= { loginBannerTable 1 }`

LoginBannerEntry ::= SEQUENCE {

loginBannerLineNumber Integer32,

loginBannerText DisplayString }

loginBannerLineNumber

Syntax	Integer32 (1..25)
Max-Access	not-accessible
Status	current
Description	The line number of this line in the login banner. Values range from 1 to the value of loginBannerLines.
Sequence	`::= { loginBannerEntry 1 }`

loginBannerText

Syntax	DisplayString (SIZE (0..80))
Max-Access	read-write
Status	current
Description	One line of text making up the login banner. The text may contain 0 to 80 characters. End-of-line characters (CR, LF) are not included. Blank lines must contain at least one space character. A zero length line terminates the prompt and deletes any following lines.
Sequence	`::= { loginBannerEntry 2 }`

Flash Memory Operations

This section is supported on all Nishan products.

nishanFlash OBJECT IDENTIFIER ::= { nishanMgmt 4 }

flashObject OBJECT IDENTIFIER ::= { nishanFlash 1 }

Information regarding locations found the flash devices

flashImageTable

Syntax	SEQUENCE OF FlashImageEntry
Max-Access	not-accessible
Status	current
Description	Table entries for Flash Locations
Sequence	`::= { flashObject 1 }`

flashImageEntry

Syntax	FlashImageEntry
Max-Access	not-accessible
Status	current
Description	Table entries for Flash Locations
	INDEX {flashImageIndex}
Sequence	`::= { flashImageTable 1 }`
	FlashImageEntry ::= SEQUENCE {
	flashImageIndex INTEGER,
	flashImageVersion DisplayString,
	flashImageCheckSum INTEGER,
	flashImageSize INTEGER,
	flashImageDate DisplayString,
	flashImageStatus INTEGER }

flashImageIndex

Syntax	INTEGER (1..4)
Max-Access	read-only
Status	current
Description	The index of this flash image.
Sequence	`::= { flashImageEntry 1 }`

flashImageVersion

Syntax	DisplayString (SIZE (0..32))
Max-Access	read-only
Status	current
Description	Version of the image downloaded into flash.
Sequence	`::= { flashImageEntry 2 }`

flashImageCheckSum

Syntax	INTEGER (0..2147483647)
Max-Access	read-only
Status	current
Description	Check sum of the image file downloaded.
Sequence	`::= { flashImageEntry 3 }`

flashImageSize

Syntax	INTEGER (0..16777215)
Max-Access	read-only
Status	current
Description	Size of the image file downloaded.
Sequence	`::= { flashImageEntry 4 }`

flashImageDate

Syntax	DisplayString (SIZE(8))
Max-Access	read-only
Status	current
Description	Time of image file creation.
Sequence	`::= { flashImageEntry 5 }`

flashImageStatus

Syntax	INTEGER (0..31)
Max-Access	read-write
Status	current
Description	If 'active' is on, then the value of 'deleted', 'invalidCRC', and 'valid' are not applicable. If 'active' is off, then the value of 'deleted', 'invalidCRC' and 'valid' must be exclusive-OR: only one is on at any time. See PortUsageType table for converting from BITS to INTEGER. 1 deleted

2 invalidCRC

3 valid

4 active

DEFVAL {0}

Sequence `::= { flashImageEntry 6 }`

File Transfer Information

upgradeServer

Syntax IpAddress

Max-Access read-write

Status current

Description Host used to retrieve flash file from. The protocol is TFTP for now so this entry is indeed the IP Address of the TFTP Server.

Sequence `::= { flashObject 2 }`

flashUploadProtocol

Syntax INTEGER {tftp(1), rcp(2), http(3), ftp(4) }

Max-Access read-write

Status obsolete

Description Protocol to use to transfer files. Not implemented at this time. Only TFTP is supported currently.

DEFVAL {tftp}

Sequence `::= { flashObject 3 }`

flashUserName

Syntax DisplayString (SIZE(0..15))

Max-Access read-write

Status current

Description User name to use with ftp (if required). Ignored by the other upload protocols.

Sequence ::= { flashObject 4 }

---

flashFileName

Syntax DisplayString (SIZE(0..255))

Max-Access read-write

Status current

Description File name with path to flash image on host. File name must come with fully qualified path.

Sequence ::= { flashObject 5 }

---

downloadPercentComplete

Syntax INTEGER (0..100)

Max-Access read-only

Status current

Description Percentage complete of the download operation.

Sequence ::= { flashObject 6 }

Flash Operation Commands

---

flashDestination

Syntax INTEGER (1..4)

Max-Access read-create

Status current

Description Index into flash of which operation will be performed against.

Sequence ::= { flashObject 7 }

---

flashCommand

Syntax INTEGER { erase(1), verify(2), upload(3) }

Max-Access read-create

Status current

Description Command to send to flash.

Sequence `::= { flashObject 8 }`

flashCommandStatus

Syntax INTEGER {
inprogress(1),
completedSuccessfully(2),
invalidHostName(3),
invalidSourceName(4),
invalidOperation(5),
deviceBusy(6),
deviceOpenError(7),
deviceFull(8),
fileOpenError(9),
fileTransferError(10),
fileCheckSumError(11),
unknownError(12),
invalidFlashFile(13) }

Max-Access read-only

Status current

Description Returns current status of command.

Sequence `::= { flashObject 9 }`

flashUploadProtocolPassword

Syntax DisplayString (SIZE(0..15))

Max-Access read-write

Status current

Description Password to use with UploadProtocol (if required).

Sequence `::= { flashObject 10 }`

runtimeImageLoadedFrom

Syntax	INTEGER {flashPartition1(1), flashPartition2(2), network(3)}
Max-Access	read-only
Status	current
Description	Returns the location from where the current runtime vxWorks Image was obtained & booted.
Sequence	`::= { flashObject 11 }`

logMsgUploadCommand

Syntax	INTEGER {upload(1), erase(2)}
Max-Access	read-write
Status	current
Description	Command to upload System Log contents to server or erase current contents of System Log. This variable can only be SET; it returns a *NoSuchName* error if read. After setting this variable to 1 or 2, poll flashCommandStatus to determine the operation status.
Sequence	`::= { flashObject 12 }`

nvramLoadingCommand

Syntax	INTEGER {upload(1),download(2)}
Max-Access	read-write
Status	current
Description	Command to upload or download nvram configuration data. This variable can only be set; it returns a *NoSuchName* error if read.
Sequence	`::= { flashObject 13 }`

logMsgUsedSpace

Syntax	INTEGER (-1..100)
Max-Access	read-only
Status	current

Description Percentage of used space in error log flash. In the case of a flash error, the value of -1 is used.

Sequence `::= { flashObject 14 }`

flashImageOnNextReset

Syntax INTEGER (1..4)

Max-Access read-write

Status current

Description Flash Image which will be used on next reset. Image is defined by flashImageIndex in flashImageTable.

Sequence `::= { flashObject 15 }`

Environmental information

This section is supported on all Nishan products.

nishanEnv	OBJECT IDENTIFIER	::= { nishanMgmt 5 }
envElectrical	OBJECT IDENTIFIER	::= { nishanEnv 1 }
envMechanical	OBJECT IDENTIFIER	::= { nishanEnv 2 }
envSetup	OBJECT IDENTIFIER	::= { nishanEnv 3 }

envVoltageTable

Syntax SEQUENCE OF EnvVoltageEntry

Max-Access not-accessible

Status current

Description Table entries for the Monitored Voltages.

Sequence `::= { envElectrical 1 }`

envVoltageEntry

Syntax EnvVoltageEntry

Max-Access not-accessible

Status current

Description Voltage monitor information.

INDEX { envVoltageIndex }

Sequence `::= { envVoltageTable 1 }`

```
EnvVoltageEntry ::= SEQUENCE {
envVoltageIndex                 INTEGER,
envVoltageDescr                 DisplayString,
envVoltageOperStatus            NishanOperStatus,
envVoltageValue                 INTEGER,
envVoltageUpperThresh           INTEGER,
envVoltageLowerThresh           INTEGER}
```

---

envVoltageIndex

Syntax INTEGER (1..6)

Max-Access read-only

Status current

Description Entry Index.

Sequence `::= { envVoltageEntry 1 }`

---

envVoltageDescr

Syntax DisplayString (SIZE (0..8))

Max-Access read-only

Status current

Description Description of measurement.

Sequence `::= { envVoltageEntry 2 }`

---

envVoltageOperStatus

Syntax NishanOperStatus

Max-Access read-only

Status current

Description Voltage status.

Sequence `::= { envVoltageEntry 3 }`

---

envVoltageValue

Syntax INTEGER (0..24000)
UNITS "millivolts"
Max-Access read-only
Status current
Description Voltage in millivolts.
Sequence `::= { envVoltageEntry 4 }`

---

envVoltageUpperThresh

Syntax INTEGER (0..24000)
UNITS "millivolts"
Max-Access read-only
Status current
Description Upper threshold before trap sent.
Sequence `::= { envVoltageEntry 5 }`

---

envVoltageLowerThresh

Syntax INTEGER (0..24000)
UNITS "millivolts"
Max-Access read-only
Status current
Description Lower threshold before trap sent.
Sequence `::= { envVoltageEntry 6 }`

---

envTemperature

Syntax INTEGER (0..200)
Max-Access read-only
Status current

Description The current ambient temperature within the measured chassis.

Sequence `::= { envMechanical 2 }`

---

envTempUpperThresh

Syntax INTEGER (0..200)

Max-Access read-write

Status current

Description The upper limit of the ambient temperature.

Sequence `::= { envMechanical 3 }`

FC Switch Zone Configuration

nishanFCsw OBJECT IDENTIFIER ::= { nishanMgmt 8 }

---

fcswMaxZoneId

Syntax INTEGER (1..4094)

Max-Access read-only

Status current

Description The maximum Zone ID that is supported in SAN Router network.

Sequence `::= { nishanFCsw 1 }`

---

fcswPortsCanBelongInMaxNumZones

Syntax INTEGER (0..4094)

Max-Access read-only

Status current

Description The Maximum number of Zones that any port can be configured in a SAN Router network.

Sequence `::= { nishanFCsw 2 }`

---

fcswNewDeviceZoneStatus

Syntax INTEGER {inNoZone(0), inDefaultZone(1)}

Max-Access read-write

Status current

Description DURABLE: { inNoZone } Zone Status for new device when plugged in this switch. Either the new device will automatically go in no zone or go in default zone.

Sequence `::= { nishanFCsw 3 }`

ZONE INFO TABLE

fcswZoneInfoTable

Syntax SEQUENCE OF FcswZoneInfoEntry

Max-Access not-accessible

Status current

Description A table containing configuration information for each ZONE configured into the device by (local or network) management. All entries are permanent and will be restored after the device is reset.

Sequence `::= { nishanFCsw 7 }`

fcswZoneInfoEntry

Syntax FcswZoneInfoEntry

Max-Access not-accessible

Status current

Description Configuration information for a ZONE configured into the device by (local or network) management.

INDEX { fcswZoneInfoTag }

Sequence `::= { fcswZoneInfoTable 1 }`

FcswZoneInfoEntry ::=

SEQUENCE {

fcswZoneInfoTag	INTEGER,
fcswZoneSymbolicName	OCTET STRING,
fcswZoneRowStatus	RowStatus,

	fcswZoneBurstLimit	INTEGER32,
	fcswZoneNormalLimit	INTEGER32,
	fcswZoneType	INTEGER}

fcswZoneInfoTag

Syntax	INTEGER (0 .. 4094)
Max-Access	read-only
Status	current
Description	The ZONE-ID that refering to this ZONE.
Sequence	`::= { fcswZoneInfoEntry 1 }`

fcswZoneSymbolicName

Syntax	OCTET STRING (SIZE (0 .. 64))
Max-Access	read-create
Status	current
Description	The symbolic port/node name field may be used to contain a variable length field (from 0 to 64) that is associated with the port/node. The content and format of this field is not defined by the name server and simply reflects information provide when this object is registered by a client. If the symbolic port/node name is not registered, then the length of this field is set to zero bytes.
Sequence	`::= { fcswZoneInfoEntry 2 }`

fcswZoneRowStatus

Syntax	RowStatus	
Max-Access	read-create	
Status	current	
Description	This object indicates the status of this entry.	
	active (1),	read-write
	notInService (2),	read-write
	notReady (3),	read-only

createAndGo (4), write-only
createAndWait (5), write-only
destroy (6), write-only

Sequence `::= { fcswZoneInfoEntry 3 }`

fcswZoneBurstLimit

Syntax Integer32 (150..170000)

Max-Access read-create

Status current

Description DURABLE: The maximum bandwidth in Kbits/sec that all sessions in this zone together may use. The maximum limit should be >= the minimum limit. The useful maximum value is currently 1,000,000 for 1G IP WAN connections (iFCP). But in subsequent releases, the maximum value could be extended to support 1,700,000 for 2G FC Connections. This is a constant limit; the 'burst' in the variable name is misleading.

NOTE: In E/OSi releases earlier than v4.7, the units for this variable are MBytes/sec, with an allowed range of 1..125.

DEFVAL {1000000}.

Sequence `::= { fcswZoneInfoEntry 4 }`

fcswZoneNormalLimit

Syntax nteger32 (150..1700000)

Max-Access read-create

Status current

Description DURABLE: The minimum bandwidth in Kbits/sec that all sessions in this zone together should be allocated. The normal limit should be <= the maximum limit. The useful maximum value is currently 1,000,000 for 1G IP WAN connections (iFCP). But in subsequent releases, the maximum value could be extended to support 1,700,000 for 2G FC Connections.

If there is insufficient bandwidth for all sessions on a link, some session traffic will be queued in a manner to guarantee the minimum

bandwidth for each session. Sessions with higher minimums may experience less queuing.

If there is insufficient bandwidth to satisfy all of the configured minimums, all sessions are queued to keep all sessions as close to their minimum as practical. The term 'normal' in the variable name may be misleading -- the actual bandwidth used is general much greater than the minimum value configured here.

NOTE: In E/OSi releases earlier than 4.7, the units for this variable are MBytes/sec, with an allowed range of 1..125.

DEFVAL {1000000}

Sequence `::= { fcswZoneInfoEntry 5 }`

fcswZoneType

Syntax INTEGER (0..2147483647)

Max-Access read-only

Status current

Description A bitmap representing the type of zone. Based on the type of zone, certain actions could be taken. For example, if it is an iSNS added zone, then this zone cannot be removed or altered.

Bit 0 is the least significant bit. Current bit assignments are:

bit 0: iSNS added zone

bit 1: ISS added zone

bits 2-31: Reserved

Sequence `::= { fcswZoneInfoEntry 6 }`

FC Port Map Table

fcswNumFcPortMapEntry

Syntax INTEGER (0..65535)

Max-Access read-only

Status current

Description The current number of entries in fcswFcPortMapTable.

Sequence `::= { nishanFCsw 8 }`

---

fcswFcPortMapTable

Syntax SEQUENCE OF FcswFcPortMapEntry

Max-Access not-accessible

Status current

Description FC Port Loop table.

Sequence `::= { nishanFCsw 9 }`

---

fcswFcPortMapEntry

Syntax FcswFcPortMapEntry

Max-Access not-accessible

Status current

Description Entry containing info. for a particular port.

INDEX { fcswFcAddressId }

Sequence `::= { fcswFcPortMapTable 1 }`

FcswFcPortMapEntry ::= SEQUENCE {

fcswFcAddressId	FCIDtype,
fcswFcNsPortSymb	OCTET STRING,
fcswFcNsPortWwn	WWNtype,
fcswFcPortType	INTEGER,
fcswNLPortEnable	INTEGER}

---

fcswFcAddressId

Syntax FCIDtype

Max-Access read-only

Status current

Description Fibre Channel ID for entry.

Sequence `::= { fcswFcPortMapEntry 1 }`

fcswFcNsPortSymb

Syntax	OCTET STRING(SIZE(0..255))
Max-Access	read-only
Status	current
Description	The object identifies the contents of a Symbolic Name of the port entry.
Sequence	`::= { fcswFcPortMapEntry 2 }`

fcswFcNsPortWwn

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	This object identifies the Fibre Channel World_wide Name of the port entry.
Sequence	`::= { fcswFcPortMapEntry 3 }`

fcswFcPortType

Syntax	INTEGER { flport(1), fport(2), nlport(3), nport(4), alport(5), lport(6) }
Max-Access	read-only
Status	current
Description	Fibre Channel Port Type.
Sequence	`::= { fcswFcPortMapEntry 4 }`

fcswNLPortEnable

Syntax	INTEGER { bypass(0), enable (1) }
Max-Access	read-write
Status	current
Description	Enable/Disable Name Server Protocol.
Sequence	`::= { fcswFcPortMapEntry 5 }`

FC Storage Name Server group

fcswSNSCommSet

Syntax	INTEGER {l2Brdcst(1), multicast(2), unicast(3)}
Max-Access	read-write
Status	current
Description	The type of communication to be used for SNS discovery. All SAN Routers in the mSAN must use the same communication type for proper operation. This variable is ignored by the Eclipse Model 1620. The value unicast(3) is not supported. For software releases before 4.1, changes to this variable take effect the next time the SAN Router is reset. For software release 4.1 and later, changes take effect immediately. In software versions 4.6 and later, the only supported value is l2Brdcst(1). Although the enumerated value name is 'l2Brdcst', SNS uses layer 3 (IP) subnet broadcast.
Sequence	`::= { nishanFCsw 10 }`

fcswSNSCommCurrent

Syntax	INTEGER {l2Brdcst(1), multicast(2), unicast(3)}
Max-Access	read-only
Status	current
Description	The type of communication currently used for SNS discovery. All SAN Routers in the mSAN must use the same communication type for proper operation. The value unicast(3) is not supported.

For software release 4.1 and later, this variable is present but not used, since it has the same value as *fcswSNSCommSet*.

In software versions 4.6 and later, the only supported value is l2Brdcst(1). Although the enumerated value name is '*l2Brdcst*', SNS uses layer 3 (IP) subnet broadcast.

Sequence `::= { nishanFCsw 11 }`

---

fcswPrimarySNSIpAddress

Syntax IpAddress

Max-Access read-only

Status current

Description The in-band IP address of the current primary SNS in this SAN.

Sequence `::= {nishanFCsw 12}`

---

fcswPrimarySNSMgtIpAddress

Syntax IpAddress

Max-Access read-only

Status current

Description The out-of-band/management IP address of the current primary SNS in this SAN.

Sequence `::= {nishanFCsw 13}`

---

fcswSNSLocalRoleSet

Syntax INTEGER { server(0), client(1) }

Max-Access read-write

Status current

Description Indicates the role that the local SNS entity will take after the next reset of the switch. server: the switch will participate in primary server election on next reboot client: the switch will not participate in primary server election on next reboot.

Sequence `::= {nishanFCsw 14 }`

fcswSNSRoleStatus

Syntax	INTEGER { server(0), client(1), serverAsClient(2) }
Max-Access	read-only
Status	current
Description	The current Name Server mode of this device. server: the switch is elected as a primary server; client: the switch did not participate in primary server election and is currently a client; server-as-client: the switch participated in primary server election but was not elected, and is currently acting as a client.
Sequence	`::= {nishanFCsw 15 }`

Naming Service's Port Table

fcswNsPortTable

Syntax	SEQUENCE OF FcswNsPortEntry
Max-Access	not-accessible
Status	current
Description	The SNS Port table for this SAN.
Sequence	`::= { nishanFCsw 16 }`

fcswNsPortEntry

Syntax	FcswNsPortEntry
Max-Access	not-accessible
Status	current
Description	An entry of the local Name Server database. INDEX { fcswNsPortWwn }
Sequence	`::= { fcswNsPortTable 1 }`

```
FcswNsPortEntry ::= SEQUENCE {
fcswNsPortWwn                   WWNtype,
fcswNsPortID                    FCIDtype,
fcswNsPortType                  INTEGER,
fcswNsPortSymb                  OCTET STRING,
fcswNsNodeWwn                   WWNtype,
fcswNsNodeSymb                  OCTET STRING,
fcswNsNodeIPA                   OCTET STRING,
fcswNsNodeIpAddress             OCTET STRING,
fcswNsPortIpAddress             OCTET STRING,
fcswNsFcCos                     INTEGER,
fcswNsFc4                       OCTET STRING,
fcswNsFabricPortWwn             WWNtype,
fcswNsDeviceZoneMap             OCTET STRING,
fcswNsDevicePriority            INTEGER,
fcswNsRmtDevFlg                 INTEGER,
fcswNsDeviceType                Integer32,
fcswNsParentNodeWwn             WWNtype,
fcswNsFc4Features               OCTET STRING}
```

fcswNsPortWwn

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The WorldWideName of the port as defined in FC-GS-3.
Sequence	`::= { fcswNsPortEntry 1 }`

fcswNsPortID

Syntax	FCIDtype
Max-Access	read-only
Status	current

Description The Port ID of the port in this SAN as used in mFCP/iFCP.

Sequence `::= { fcswNsPortEntry 2 }`

fcswNsPortType

Syntax INTEGER {
unknown (0),
nPort (1),
nlPort (2),
fNlPort (3),
nSoIPPort (16), x'10'
fPort (129), x'81'
flPort (130), x'82'
rPort (132), x'84'
bPort (133), x'85'
lport (134), x'86'
fEtherPort (144), x'90'
fEtherGtwy (145), x'91'
omniPort (146), x'92'
fIscsiGtwy (147), x'93,
unknownEnd (255)

Max-Access read-only

Status current

Description The identifies the type of port: N_Port, NL_Port, etc., for this entry. The port types are consistent with those defined in FC-GS-3, with extensions. The extensions are: fEtherPort, fEtherGtwy, omniPort, nSoIPPort and fIscsiGtwy.

Sequence `::= { fcswNsPortEntry 3 }`

fcswNsPortSymb

Syntax OCTET STRING(SIZE(0..255))

Max-Access read-write

Status current

Description — The Symbolic Name of the port as defined in FC-GS-3. This is a variable-length text-based description of up to 255 bytes, that is associated with the device port in the network. This field is normally provided during device registration. However, a network management application can update this field as required.

Sequence — `::= { fcswNsPortEntry 4 }`

---

fcswNsNodeWwn

Syntax — WWNtype

Max-Access — read-only

Status — current

Description — The WorldWideName of the node as defined in FC-GS-3.

Sequence — `::= { fcswNsPortEntry 5 }`

---

fcswNsNodeSymb

Syntax — OCTET STRING (SIZE(0..255))

Max-Access — read-only

Status — current

Description — The object identifies the contents of a Symbolic Name of the node associated with the entry.

Sequence — `::= { fcswNsPortEntry 6 }`

---

fcswNsNodeIPA

Syntax — OCTET STRING (SIZE(8))

Max-Access — read-only

Status — current

Description — The object identifies the Initial Process Associator of the node for the entry as defined in FC-GS-2.

Sequence — `::= { fcswNsPortEntry 7 }`

---

fcswNsNodeIpAddress

Syntax — OCTET STRING (SIZE(16))

Max-Access read-only

Status current

Description The IP address of the node as defined in FC-GS-3. The format of the address is in IPv6. When an IPv4 value is contained in this field, the most significant 12 bytes are set to 0x00. By convention, this IP Address can be used as the management IP Address for the device.

Sequence `::= { fcswNsPortEntry 8 }`

---

fcswNsPortIpAddress

Syntax OCTET STRING (SIZE(16))

Max-Access read-only

Status current

Description The IP address of the node as defined in FC-GS-3. The format of the address is in IPv6. When an IPv4 value is contained in this field, the most significant 12 bytes are set to 0x00.

Sequence `::= { fcswNsPortEntry 9 }`

---

fcswNsFcCos

Syntax INTEGER {

classUnknown (0),

classF (1),

class1 (2),

classF1 (3),

class2 (4),

classF2 (5),

class12 (6),

classF12 (7),

class3 (8),

classF3 (9),

class13 (10),

classF13 (11),

class23 (12),

classF23 (13),

class123 (14),

classF123 (15) }

Max-Access read-only

Status current

Description The object identifies the class of services supported by the port. The value is a bit-map defined as follows:

o bit 0 is class F,

o bit 1 is class 1,

o bit 2 is class 2,

o bit 3 is class 3,

o bit 4 is class 4, etc.

Sequence `::= { fcswNsPortEntry 10 }`

---

fcswNsFc4

Syntax OCTET STRING (SIZE (32))

Max-Access read-only

Status current

Description The object identifies the FC-4s supported by the port as defined in FC-GS-3.

Sequence `::= { fcswNsPortEntry 11 }`

---

fcswNsFabricPortWwn

Syntax WWNtype

Max-Access read-only

Status current

Description The object identifies the Fibre Channel World Wide Name of the associated switch port as defined in FC-GS-3.

Sequence `::= { fcswNsPortEntry 13}`

fcswNsDeviceZoneMap

Syntax	OCTET STRING (SIZE (1..512))
Max-Access	read-write
Status	current
Description	DURABLE: { 0:all } The ZONE-ID BIT map to which this device belongs. The total number of zone supported in SAN Router network environment is determined by fcswMaxZoneId. Each bit in this object refers to a zone id, e.g. bit 1 referring to zone 1, bit 2 referring to zone 2 etc. Number of octets in this octet string should be equal to fcswMaxZoneId/8. Only fcswPortsCanBelongInMaxNumZones number of bits should be set as fcswPortsCanBelongInMaxNumZones are the number of zones a device can belong into.
Sequence	`::= { fcswNsPortEntry 14 }`

fcswNsDevicePriority

Syntax	INTEGER (0..7)
Max-Access	read-write
Status	current
Description	DURABLE: { 1:all } Priority of this device.
Sequence	`::= { fcswNsPortEntry 15 }`

fcswNsRmtDevFlg

Syntax	INTEGER { snsLocalDevice (1), snsRemoteDevice (2) }
Max-Access	read-only
Status	current
Description	This is to differentiate remote devices from local devices. Default for this flag is 1 (SNS_LOCAL_DEVICE). Remote devices are 2 (SNS_REMOTE_DEVICE).
Sequence	`::= { fcswNsPortEntry 16}`

fcswNsDeviceType

Range removed to avoid MIB compiler errors. This should be an Unsigned32 or octet string, but the agent returns an ASN.1 INTEGER.

Syntax INTEGER (0 .. 4294967295)

Syntax Integer32

Max-Access read-only

Status current

Description This mib variable is deprecated from switch software release 3.1 and beyond. This attribute indicates the type of device that is registered. It can be used by network management to provide hints to the user what type of functions can be performed on this device.

Byte 0 gives SCSI device type Identifies the type of SCSI Device

Code	Description
00h	Direct-access device.
01h	Sequential-access device.
02h	Printer device.
03h	Processor device.
04h	WORM device.
05h	CD-ROM device.
06h	Scanner device.
07h	Optical Memory device.
08h	Medium Changer device.
09h	Communications device
0Ah - 1Eh	Reserved.
1Fh	Unknown or no device type.

Byte 1 gives FC Port Type;

0 INVALID

1 PRIVATE

2 PUBLIC

3 ALIEN

Byte 2 is a bit-map that indicates device role.

Both Bits 0,1 = 0 means device role is UNKNOWN.

Bit 0 = 1 means role is TARGET

Bit 1 = 1 means role is INITIATOR

Both Bits 0,1 = 1 means role is TARGET AND INITIATOR

Byte 3 is reserved.

Sequence `::= { fcswNsPortEntry 17 }`

---

fcswNsParentNodeWwn

Syntax WWNtype

Max-Access read-only

Status current

Description The object identifies the node World Wide Name of the associated node if exists.

Sequence `::= { fcswNsPortEntry 18}`

---

fcswNsFc4Features

Syntax OCTET STRING (SIZE (128))

Max-Access read-only

Status current

Description The object identifies the FC-4 Features supported by the port as defined in FC-GS-3.

For FC4 type FCP-2 (defined in FC-PH as type 08h), the FC4 feature bits are defined as follows:

Bit	Interpretation
3	Reserved
2	Reserved
1	1 = FCP Initiator function supported, 0 = not supported
0	1 = FCP Target function supported, 0 = not supported

Sequence `::= { fcswNsPortEntry 19 }`

AL Port Map Table

fcswALPortMappingTable

Syntax	SEQUENCE OF FcswALPortMappingEntry
Max-Access	not-accessible
Status	current
Description	Arbitrated Loop port virtual mapping table.
Sequence	`::= { nishanFCsw 17 }`

fcswALPortMappingEntry

Syntax	FcswALPortMappingEntry
Max-Access	not-accessible
Status	current
Description	Entry containing mapping information for an Arbitrated Loop port. INDEX { fcswALFcAddressId }

Sequence	`::= { fcswALPortMappingTable 1 }`
	FcswALPortMappingEntry ::= SEQUENCE {
	fcswALFcAddressId FCIDtype,
	fcswALVirtualFcAddressId FCIDtype}

fcswALFcAddressId

Syntax	FCIDtype
Max-Access	read-only
Status	current
Description	Arbitrated Loop port real FC Address ID.
Sequence	`::= { fcswALPortMappingEntry 1 }`

fcswALVirtualFcAddressId

Syntax	FCIDtype

Max-Access read-only
Status current
Description Arbitrated Loop port virtual FC Address ID.
Sequence `::= { fcswALPortMappingEntry 2 }`

Naming Service's Node Table

fcswNsNodeTable

Syntax SEQUENCE OF FcswNsNodeEntry
Max-Access not-accessible
Status current
Description The SNS Node table for this SAN.
Sequence `::= { nishanFCsw 18 }`

fcswNsNodeEntry

Syntax FcswNsNodeEntry
Max-Access not-accessible
Status current
Description An entry of the Node in the Name Server database.
INDEX { fcswNsNodeWwn }
Sequence `::= { fcswNsNodeTable 1 }`

```
FcswNsNodeEntry ::= SEQUENCE {
fcswNtNsNodeWwn          WWNtype,
fcswNtNsNodeIpAddress    OCTET STRING,
fcswNtNsNodeSymb         OCTET STRING,
fcswNtNsNodeIPA          OCTET STRING}
```

fcswNtNsNodeWwn

Syntax WWNtype
Max-Access read-only

Status	current
Description	The object identifies the Fibre Channel World Wide Name of the associated node as defined in FC-GS-3.
Sequence	`::= { fcswNsNodeEntry 1 }`

fcswNtNsNodeIpAddress

Syntax	OCTET STRING (SIZE(16))
Max-Access	read-only
Status	current
Description	The IP address of the node as defined in FC-GS-3. The format of the address is in IPv6. When an IPv4 value is contained in this field, the most significant 12 bytes are set to 0x00. By convention, this IP Address can be used as the management IP Address for the device.
Sequence	`::= { fcswNsNodeEntry 2 }`

fcswNtNsNodeSymb

Syntax	OCTET STRING (SIZE(0..255))
Max-Access	read-write
Status	obsolete
Description	The Symbolic Name of the node as defined in FC-GS-3. This is a variable-length text-based description of up to 255 bytes, that is associated with the device node in the network. This field is normally provided during device registration. However, a network management application can update this field as required. **NOTE:** This variable is no longer supported in current products. Use fcswNsNodeSymb instead.
Sequence	`::= { fcswNsNodeEntry 3 }`

fcswNtNsNodeIPA

Syntax	OCTET STRING (SIZE(8))
Max-Access	read-only
Status	Obsolete

Description The object identifies the Initial Process Associator of the node for the entry as defined in FC-GS-2.

Sequence `::= { fcswNsNodeEntry 4 }`

---

fcpPortTable

Syntax SEQUENCE OF FcpPortEntry

Max-Access not-accessible

Status current

Description This table gives/stores registered SCSI FCP targets properties.

Sequence `::= { nishanFCsw 19 }`

---

fcpPortEntry

Syntax FcpPortEntry

Max-Access not-accessible

Status current

Description This table gives/stores registered SCSI FCP targets properties.

INDEX { fpPortName}

Sequence `::= { fcpPortTable 1 }`

FcpPortEntry ::= SEQUENCE {

fpPortName	WWNtype,
fpRemovable	TruthValue,
fpAnsiVersion	INTEGER,
fpVendorId	DisplayString,
fpProductId	DisplayString,
fpProductRevision	DisplayString,
fpLogicalBlocks	INTEGER,
fpBlockLength	INTEGER,
fpCapacity	INTEGER,
fpDeviceType	Integer32

fpPortName

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	8-byte World Wide Portname.
Sequence	`::= {fcpPortEntry 1 }`

fpRemovable

Syntax	TruthValue
Max-Access	read-only
Status	current
Description	Identifies that this device has removable media. False. Media is not removable. True. Media is removable.
Sequence	`::= { fcpPortEntry 2 }`

fpAnsiVersion

Syntax	INTEGER (0..7)
Max-Access	read-only
Status	current
Description	Indicates the implement ANSI version of this device. 0 might or might not comply to ANSI standards. 1 complies to ANSI X3.131-1966 (SCSI-I). 2 compiles to ANSI ????? (SCSI-II). 3 - 7 reserved.
Sequence	`::= { fcpPortEntry 3 }`

fpVendorId

Syntax	DisplayString

Max-Access	read-only
Status	current
Description	Indicates the vendor ID.
Sequence	`::= { fcpPortEntry 4 }`

fpProductId

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	Indicates the Product ID.
Sequence	`::= { fcpPortEntry 5 }`

fpProductRevision

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	Indicates the Product Revision.
Sequence	`::= { fcpPortEntry 6 }`

fpLogicalBlocks

Syntax	INTEGER (0..2147483647)
Max-Access	read-only
Status	current
Description	A 32-bit value that represents the total number of logical blocks for this device. Octet 1 is the LSB, octet 4 is the MSB.
Sequence	`::= { fcpPortEntry 7 }`

fpBlockLength

Syntax	INTEGER (0..2147483647)
Max-Access	read-only

Status current

Description A 32-bit value that represents the size of the logical block for this device. Octet 1 is the LSB, octet 4 is the MSB.

Sequence `::= { fcpPortEntry 8 }`

fpCapacity

Syntax INTEGER (0..2147483647)

UNITS "megabytes"

Max-Access read-only

Status current

Description A value that represents the capacity of the device in megabytes. One MegaByte equals to 1,048,576 when calculating this value.

Sequence `::= { fcpPortEntry 9 }`

fpDeviceType

NOTE: Range removed to avoid MIB compiler errors. This should be an Unsigned32 or octet string, but the agent returns an ASN.1 INTEGER.

Syntax INTEGER (0 .. 4294967295)

Syntax Integer32

Max-Access read-only

Status current

Description Byte 0 gives SCSI device type Identifies the type of SCSI Device.

Code	Description
00h	Direct-access device.
01h	Sequential-access device.
02h	Printer device.
03h	Processor device.
04h	WORM device.
05h	D-ROM device.
06h	Scanner device.

07h	Optical Memory device.
08h	Medium Changer device.
09h	Communications device
0Ah - 1Eh	Reserved.
1Fh	Unknown or no device type.

Byte 1 gives FC Port Type;

0 INVALID

1 PRIVATE

2 PUBLIC

3 ALIEN

Byte 2 is a bit-map that indicates device role;

Both Bits 0,1 = 0 means device role is UNKNOWN.

Bit 0 = 1 means role is TARGET

Bit 1 = 1 means role is INITIATOR

Both Bits 0,1 = 1 means role is TARGET AND INITIATOR

Byte 3 is reserved.

Sequence `::= { fcpPortEntry 10 }`

FC Name Server Configuration

fcswSNSCommPortSet

Syntax INTEGER (0..65535)

Max-Access read-write

Status current

Description For software versions before 4.1, this variable indicates the UDP/TCP port that the local SNS entity will use after the next reset of the switch.

For software version 4.1 and later, this variable indicates the UDP/TCP port currently used by SNS. Any changes take effect immediately.

All switches in the SAN must use the same port number for proper operation.

Sequence `::= {nishanFCsw 20 }`

---

fcswSNSCommPortCurrent

Syntax INTEGER (0..65535)

Max-Access read-only

Status current

Description The current UDP port being used for name server communication. For software version 4.1 and later, this variable returns the same value as fcswSNSCommPortSet.

Sequence `::= {nishanFCsw 21 }`

---

fcswSNSL3McastGrpSet

Syntax IpAddress

Max-Access read-write

Status current

Description For software versions before 4.1, this variable indicates the multicast group that the local SNS entity will use after the next reset of the switch.

For software version 4.1 through 4.5, this variable indicates the multicast group currently being used by SNS. Any changes take effect immediately.

For software version 4.6 and later, this variable is ignored because multicast is no longer supported. All switches in the SAN must use the same multicast group for proper functionality. The group is only used if multicast communication has been set.

Sequence `::= {nishanFCsw 22 }`

---

fcswSNSL3McastGrpCurrent

Syntax IpAddress

Max-Access read-only

Status current

Description The current multicast group being used for name server communication. This group is only valid if multicast communication has been set.

For software version 4.1 or later, this variable returns the same value as fcswSNSL3McastGrpSet.

Sequence `::= {nishanFCsw 23 }`

---

fcswNumSNSPorts

Syntax INTEGER (0..65535)

Max-Access read-only

Status current

Description The current number of entries in Name Server's port table.

Sequence `::= { nishanFCsw 24 }`

---

fcswNumSNSNodes

Syntax INTEGER (0..65535)

Max-Access read-only

Status current

Description The current number of entries in Name Server's node table.

Sequence `::= { nishanFCsw 25 }`

---

fcswNumSNSZones

Syntax INTEGER (0..65535)

Max-Access read-only

Status current

Description The current total number of entries in SNS zone table.

Sequence `::= { nishanFCsw 26}`

SNS Version info

fcswSNSrevInfo OBJECT IDENTIFIER ::= { nishanFCsw 30 }

fcswSNSrevLclVer

Syntax	INTEGER (0 .. 1024)
Max-Access	read-only
Status	current
Description	The version of the local SNS on this switch. One of the switches with the highest SNS version, and then SNS priority, will become the primary SNS for the network. The SNS version is fixed for a specific firmware release.
Sequence	`::= { fcswSNSrevInfo 1 }`

fcswSNSrevLclPriority

Syntax	INTEGER (0 .. 128)
Max-Access	read-write
Status	current
Description	DURABLE: The priority of the local SNS on this switch. One of the switches with the highest SNS version, and then SNS priority, will become the primary SNS for the network. Higher values indicate higher priority. The SNS Priority is configurable by the user.
Sequence	`::= { fcswSNSrevInfo 2 }`
ZONE SET INFO	fcswSNSzsInfo OBJECT IDENTIFIER ::= { nishanFCsw 31 }

fcswSNSzstTable

Syntax	SEQUENCE OF FcswSNSzstEntry
Max-Access	not-accessible
Status	current
Description	A table containing configuration information for each ZONE SET configured into the device by (local or network) management. All entries are permanent once saved to flash and will be restored after the device is reset. Zonesets are not currently supported. This table is empty for current products.

Sequence `::= { fcswSNSzsInfo 1 }`

fcswSNSzstEntry

Syntax FcswSNSzstEntry

Max-Access not-accessible

Status current

Description Configuration information for a ZONE SET configured into the device by (local or network) management.

INDEX { fcswSNSzstZoneSetID }

Sequence `::= { fcswSNSzstTable 1 }`

FcswSNSzstEntry ::=

SEQUENCE {

fcswSNSzstZoneSetID	INTEGER,
fcswSNSzstSymbolicName	OCTET STRING,
fcswSNSzstStatus	INTEGER,
fcswSNSzstZoneMap	OCTET STRING,
fcswSNSzstRowStatus	RowStatus}

fcswSNSzstZoneSetID

Syntax INTEGER (1..64)

Max-Access read-only

Status current

Description DURABLE: { 0:all }

The ZONE SET ID that refers to this ZONE SET.

Sequence `::= { fcswSNSzstEntry 1 }`

fcswSNSzstSymbolicName

Syntax OCTET STRING (SIZE (0 .. 64))

Max-Access read-create

Status current

Description DURABLE: { 0:all }

The Zone Set Symbolic Name field may be used to contain a variable length field (from 0 to 64) that is associated with the Zone Set. If the name is not registered, then the length of this field is set to zero bytes.

Sequence `::= { fcswSNSzstEntry 2 }`

---

fcswSNSzstStatus

Syntax INTEGER { disabled (0), enabled (1) }

Max-Access read-create

Status current

Description DURABLE: { 0:all }

ZONE SET Status indicates whether the Zone Set is currently enabled.

Sequence `::= { fcswSNSzstEntry 3 }`

---

fcswSNSzstZoneMap

Syntax OCTET STRING (SIZE (0 .. 512))

Max-Access read-create

Status current

Description DURABLE: { 0:all }

The ZONE SET ZONE MAP is a bitmap that indicates which Zones are members of this Zone Set.

Sequence `::= { fcswSNSzstEntry 4 }`

---

fcswSNSzstRowStatus

Syntax RowStatus

Max-Access read-create

Status current

Description	This object indicates the status of this entry.
	active (1), read-write
	notInService (2), read-write
	notReady (3), read-only
	createAndGo (4), write-only
	createAndWait (5), write-only
	destroy (6), write-only
Sequence	`::= { fcswSNSzstEntry 5 }`

iSCSI LUN Mapping and Masking

This section is supported on Eclipse switches for software release 3.0 and above.

fcswSNSLunMapping

Syntax	INTEGER {disable(0), enable(1)}
Max-Access	read-write
Status	current
Description	Indicates whether LUN Masking/Mapping is enabled or disabled.
Sequence	`::= {nishanFCsw 32 }`

fcswLunsTable

Syntax	SEQUENCE OF FcswLunsEntry
Max-Access	not-accessible
Status	current
Description	Contains status parameters specific to LUNs.
Sequence	`::= { nishanFCsw 33 }`

fcswLunsEntry

Syntax	FcswLunsEntry
Max-Access	not-accessible

Status	current
Description	Information for Nishan FC Targets.
	INDEX {fcswTargetWWN}
Sequence	`::= {fcswLunsTable 1}`
	FcswLunsEntry ::=
	SEQUENCE {
	fcswTargetWWN WWNtype,
	fcswNumberOfLuns INTEGER }

fcswTargetWWN

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The Port WWN for this FC target.
Sequence	`::= { fcswLunsEntry 1 }`

fcswNumberOfLuns

Syntax	INTEGER (0..255)
Max-Access	read-only
Status	current
Description	This object contains the number of Luns for this target.
Sequence	`::= { fcswLunsEntry 2 }`

fcswLunInfoTable

fcswLunInfoTable

Syntax	SEQUENCE OF FcswLunInfoEntry
Max-Access	not-accessible
Status	current
Description	Contains status parameters specific to LUNs.

Sequence `::= { nishanFCsw 34 }`

fcswLunInfoEntry

Syntax FcswLunInfoEntry

Max-Access not-accessible

Status current

Description Information for the Nishan SNS LUNs.

INDEX { fcswfcTargetWWN,fcswLunIndex }

Sequence `::= {fcswLunInfoTable 1}`

FcswLunInfoEntry ::=

SEQUENCE {

fcswfcTargetWWN	WWNtype,
fcswLunIndex	INTEGER,
fcswLunWWN	WWNtype,
fcswLunVendorId	DisplayString,
fcswLunProductId	DisplayString,
fcswLunProductRev	DisplayString,
fcswLunLogicalBlocks	Integer32,
fcswLunLBLength	Integer32,
fcswLunCapacity	Integer32,
fcswLunDeviceType	Integer32,
fcswLunRemovable	TruthValue,
fcswLunScsiVersion	Integer32,
fcswLunNumber	Integer32 }

fcswfcTargetWWN

Syntax WWNtype

Max-Access read-only

Status current

Description The Port WWN for this FC target.

Sequence `::= { fcswLunInfoEntry 1 }`

fcswLunIndex

Syntax	INTEGER (1..256)
Max-Access	read-only
Status	current
Description	The sequential index for the table.
Sequence	`::= {fcswLunInfoEntry 2}`

fcswLunWWN

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The Lun WWN, if available for this LUN.
Sequence	`::= { fcswLunInfoEntry 3 }`

fcswLunVendorId

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	The Vendor ID for this LUN.
Sequence	`::= { fcswLunInfoEntry 4 }`

fcswLunProductId

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	The Product ID for this LUN.
Sequence	`::= { fcswLunInfoEntry 5 }`

fcswLunProductRev

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	The Product Revision for this LUN.
Sequence	`::= { fcswLunInfoEntry 6 }`

fcswLunLogicalBlocks

Syntax	Integer32
Max-Access	read-only
Status	current
Description	A 32-bit value that represents the total number of logical blocks for this device.
Sequence	`::= { fcswLunInfoEntry 7}`

fcswLunLBLength

Syntax	Integer32
Max-Access	read-only
Status	current
Description	A 32-bit value that represents the size of the logical block for this device, in bytes. Typical values are 512, 1024, 2048...
Sequence	`::= { fcswLunInfoEntry 8}`

fcswLunCapacity

Syntax	Integer32 UNITS "megabytes"
Max-Access	read-only
Status	current
Description	A value that represents the capacity of the device in megabytes. One MegaByte equals to 1,048,576 when calculating this value.

Sequence ::= { fcswLunInfoEntry 9}

fcswLunDeviceType

Syntax Integer32

Max-Access read-only

Status current

Description Byte 0 gives SCSI device type Identifies the type of SCSI Device:

Code	Description
00h	Direct-access device
01h	Sequential-access device
02h	Printer device
03h	Processor device
04h	Write-once device
05h	CD-ROM device
06h	Scanner device
07h	Optical Memory device
08h	Medium Changer device
09h	Communications device
0Ah - 0Bh	Graphics arts pre-press devices
0Ch	Storage array controller (RAID)
0Dh	Enclosure services device
0Eh	Simp. direct-access dev (mag disk)
0Fh	Optical card reader/writer device
10h	Reserved
11h	Object-based Storage Device
12h - 1Eh	Reserved
1Fh	Unknown or no device type

Byte 1 gives FC Port Type;

0 INVALID

1 PRIVATE

2 PUBLIC

3 ALIEN

Byte 2 is a bit-map that indicates device role;

Both Bits 0,1 = 0 means device role is UNKNOWN.

Bit 0 = 1 means role is TARGET

Bit 1 = 1 means role is INITIATOR

Both Bits 0,1 = 1 means role is TARGET AND INITIATOR

Byte 3 is reserved.

Sequence `::= { fcswLunInfoEntry 10 }`

fcswLunRemovable

Syntax TruthValue

Max-Access read-only

Status current

Description Identifies that this device has removable media.

False. Media is not removable.

True. Media is removable.

Sequence `::= { fcswLunInfoEntry 11 }`

fcswLunScsiVersion

Syntax Integer32 (0..255)

Max-Access read-only

Status current

Description The SCSI Version supported as reported by the Inquiry command as defined in SCSI Primary Commands - 2 (SPC-2).

Code	Description
00h	conformance not specified
01h	Obsolete (SCSI-I)
02h	(SCSI-II)
03h	(SCSI-III)
04h	Complies to SPC-2

82h (SCSI-II) & ISO/IEC 9316:1995

83h (SCSI-III) & ISO/IEC 9316:1995

84h Complies to SPC-2 & ISO/IEC 9316:1995

Other values are also defined. The above list shows current pertinent values.

Sequence `::= { fcswLunInfoEntry 12 }`

fcswLunNumber

Syntax Integer32

Max-Access read-only

Status current

Description The physical LUN number that contains the 2 byte LUN Field. The upper byte has the address field, which might be used in older RAID systems.

The lower byte contains the lun value itself. For ex. 0x40 is the upper byte for certain RAID systems to indicate no lun hierarchy.

Sequence `::= { fcswLunInfoEntry 13 }`

fcswLunMappingTable

fcswLunMappingTable

Syntax SEQUENCE OF FcswLunMappingEntry

Max-Access not-accessible

Status current

Description Contains status parameters specific to LUNs.

Sequence `::= { nishanFCsw 35 }`

fcswLunMappingEntry

Syntax FcswLunMappingEntry

Max-Access not-accessible

Status current

Description	Information for the Nishan SNS LUNs.
	INDEX { fcswfcDeviceWWN, fcswZoneId, fcswPLunIndex }
Sequence	`::= {fcswLunMappingTable 1}`
	FcswLunMappingEntry ::=
	SEQUENCE {
	fcswfcDeviceWWN WWNtype,
	fcswZoneId INTEGER,
	fcswPLunIndex INTEGER,
	fcswVLuNumber INTEGER,
	fcswLunMaskedStatus INTEGER}

fcswfcDeviceWWN

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The Port WWN for this FC target.
Sequence	`::= { fcswLunMappingEntry 1 }`

fcswZoneId

Syntax	INTEGER (1..512)
Max-Access	read-only
Status	current
Description	This value is the zone id for which this mapping is done.
Sequence	`::= {fcswLunMappingEntry 2}`

fcswPLunIndex

Syntax	INTEGER (1..256)
Max-Access	read-only
Status	current
Description	This is the index of the physical LUN from the lun info table.

Sequence ::= {fcswLunMappingEntry 3}

fcswVLuNumber

Syntax INTEGER (0..65535)

Max-Access read-write

Status current

Description This is the assigned Virtual Lun Number for the FC lun.

Sequence ::= {fcswLunMappingEntry 4}

fcswLunMaskedStatus

Syntax INTEGER { visible(1), invisible(2), delete(3) }

Max-Access read-write

Status current

Description This object indicates the status of this LUN, whether it is masked or not.

visible (1),	not-masked
invisible (2),	masked
delete (3)	delete the mapping for this zone.

If the lun mapping becomes invalid, then the existing value is OR'd by 8. i.e. 4th bit is set to 1. For. ex, if the value is 10 (1010), then this lun was invisible before the lun map became in-valid.

One of the reasons that the lun map could be invalid is if the lun registration fails after a device is plugged-in.

Sequence ::= { fcswLunMappingEntry 5 }

fcswSnsClientTable

fcswSnsClientCount

Syntax Integer32 (0..255)

Max-Access read-only

Status current

Description — The number of entries in fcswSnsClientTable. If this SAN Router is the primary SNS (MIB variable fcswSNSRoleStatus == 0), then fcswSnsClientCount is the number of SNS clients registered (0 or more) and fcswSnsClientTable contains one entry for each client.

If this SAN Router is not the primary SNS, then fcswSnsClientCount is 1, and fcswSnsClientTable contains 1 entry for the local client. For software releases and hardware models that don't support mFCP, fcswSnsClientCount is always 0.

Sequence — `::= { nishanFCsw 36 }`

---

fcswSnsClientTable

Syntax — SEQUENCE OF FcswSnsClientEntry

Max-Access — not-accessible

Status — current

Description — Information on SNS clients. If this SAN Router is the primary SNS (MIB variable fcswSNSRoleStatus == 0), then fcswSnsClientTable contains one entry for each remote client. If this SAN Router is not the primary SNS, then fcswSnsClientTable contains 1 entry for the local client.

For software releases and hardware models that don't support mFCP, this table is always empty.

Sequence — `::= { nishanFCsw 37 }`

---

fcswSnsClientEntry

Syntax — FcswSnsClientEntry

Max-Access — not-accessible

Status — current

Description — Information for one SNS client.

INDEX { fcswSnsClientIndex }

Sequence — `::= {fcswSnsClientTable 1}`

FcswSnsClientEntry ::=

SEQUENCE {	
fcswSnsClientIndex	Integer32,
fcswSnsClientInbandAddrType	InetAddressType,
fcswSnsClientInbandAddr	InetAddress,
fcswSnsClientMgmtPortAddrType	InetAddressType,
fcswSnsClientMgmtPortAddr	InetAddress,
fcswSnsClientRegDate	DateAndTime,
fcswSnsClientConflicts	Integer32 }

fcswSnsClientIndex

Syntax	Integer32 (1..255)
Max-Access	not-accessible
Status	current
Description	An arbitrary integer to identify an SNS client. Client indexes do not need to be contiguous.
Sequence	`::= { fcswSnsClientEntry 1 }`

fcswSnsClientInbandAddrType

Syntax	InetAddressType
Max-Access	read-only
Status	current
Description	The format of the IP address in fcswSnsClientInbandAddr. Currently only IPv4(1) is supported.
Sequence	`::= { fcswSnsClientEntry 2 }`

fcswSnsClientInbandAddr

Syntax	InetAddress
Max-Access	read-only
Status	current
Description	The switch address of the SNS client. This is the mFCP address used in SNS packets.

Sequence ::= { fcswSnsClientEntry 3 }

fcswSnsClientMgmtPortAddrType

Syntax InetAddressType

Max-Access read-only

Status current

Description The format of the IP address in fcswSnsClientMgmtPortAddr. Currently only IPv4(1) is supported.

Sequence ::= { fcswSnsClientEntry 4 }

fcswSnsClientMgmtPortAddr

Syntax InetAddress

Max-Access read-only

Status current

Description The management port address of the SNS client. This address is used by management applications performing out-of-band management.

Sequence ::= { fcswSnsClientEntry 5 }

fcswSnsClientRegDate

Syntax DateAndTime

Max-Access read-only

Status current

Description The date and time when this client was registered with the primary SNS, according to the local SAN Router's clock (i.e., the clock in the SAN Router containing this MIB table).

This is the time at which configuration conflicts listed in fcswSnsClientConflicts were detected.

Sequence ::= { fcswSnsClientEntry 6 }

fcswSnsClientConflicts

Syntax Integer32

Max-Access read-only

Status current

Description A bitmap indicating configuration conflicts between this client and the primary SNS. Conflicts are detected when the primary SNS is elected or when the client joined the mSAN. The bitmap is not updated after the initial detection, so this may not be current information.

Bit 0 is the least-significant bit. Bit meanings are:

bit 0 fabric interconnect mode conflict

bit 1 fabric zone policy conflict

bit 2 cluster ID conflict

bit 3 iFCP SAN ID conflict

bit 4 default zoning mode conflict

bit 5 jumbo frames setting conflict

bit 6 LUN Mapping setting conflict

A value of 0 indicates no configuration conflicts.

Sequence `::= { fcswSnsClientEntry 7 }`

Chassis information

nishanFCswChassis OBJECT IDENTIFIER ::= { nishanMgmt 9 }

PHYSICAL Elements

fcswChasType

Syntax INTEGER {pebrine2(1), -- P2 box
others(2) -- other boxes }

Max-Access read-only

Status current

Description Chassis Type

Sequence `::= { nishanFCswChassis 1 }`

fcswChasSerialNumber

Syntax	DisplayString (SIZE (0..8))
Max-Access	read-only
Status	current
Description	Unique serial number for this chassis.
Sequence	`::= { nishanFCswChassis 2 }`

fcswChasHardwareRevision

Syntax	DisplayString (SIZE (0..20))
Max-Access	read-only
Status	current
Description	Device hardware revision level
Sequence	`::= { nishanFCswChassis 3 }`

fcswChasPartNumber

Syntax	DisplayString (SIZE (0..6))
Max-Access	read-only
Status	current
Description	Chassis Part Number.
Sequence	`::= { nishanFCswChassis 4 }`

fcswChasNumSlots

Syntax	INTEGER (1..16)
Max-Access	read-only
Status	current
Description	The number of slots in the chassis available for plug-in cards.
Sequence	`::= { nishanFCswChassis 5 }`

fcswChasNumPorts

Syntax	INTEGER (0..255)
Max-Access	read-only
Status	current
Description	The total number of ports currently installed in the chassis.
Sequence	`::= { nishanFCswChassis 6 }`

fcswChasNumFcPorts

Syntax	INTEGER (0..255)
Max-Access	read-only
Status	current
Description	The total number of Fibre Channel ports currently installed the chassis.
Sequence	`::= { nishanFCswChassis 7 }`

fcswChasNumPowerSupply

Syntax	INTEGER (0..255)
Max-Access	read-only
Status	current
Description	The total number of Power Supply currently installed in the chassis.
Sequence	`::= { nishanFCswChassis 8 }`

fcswChasNumFans

Syntax	INTEGER (0..255)
Max-Access	read-only
Status	current
Description	The total number of Fans currently installed in the chassis.
Sequence	`::= { nishanFCswChassis 9 }`

Fan

fcswChasFan OBJECT IDENTIFIER ::= { nishanFCswChassis 10 }

fcswChasFanTable

Syntax	SEQUENCE OF FcswChasFanEntry
Max-Access	not-accessible
Status	current
Description	This table contains information about Fan.
Sequence	`::= { fcswChasFan 1 }`

fcswChasFanEntry

Syntax	FcswChasFanEntry
Max-Access	not-accessible
Status	current
Description	Contains information regarding a power supply unit. INDEX { fcswChasFanIndex }

Sequence	`::= { fcswChasFanTable 1 }`
	FcswChasFanEntry ::=
	SEQUENCE {
	fcswChasFanIndex INTEGER,
	fcswChasFanDescr DisplayString,
	fcswChasFanPartNumber DisplayString,
	fcswChasFanOperStatus INTEGER,
	fcswChasFanValue INTEGER }

fcswChasFanIndex

Syntax	INTEGER(1..2)
Max-Access	read-only
Status	current
Description	An index to this fan instance.

Sequence ::= { fcswChasFanEntry 1 }

---

fcswChasFanDescr

Syntax DisplayString (SIZE (0..14))

Max-Access read-only

Status current

Description A textual description of the location of the fan.

Sequence ::= { fcswChasFanEntry 2 }

---

fcswChasFanPartNumber

Syntax DisplayString (SIZE (0..6))

Max-Access read-only

Status obsolete

Description Fan Part Number -- note: this object has been obsoleted as the part number cannot always be determined.

Sequence ::= { fcswChasFanEntry 3 }

---

fcswChasFanOperStatus

Syntax INTEGER {

unknown(1),

norma(2),

critical(3),

warning(4)

}

Max-Access read-only

Status current

Description — Actual status of the Fan:

unknown(1) -	status can not be determined.
normal(2) -	present and operating within a 20% variance of acceptable fan speeds.
critical(3)-	present but operating above a 40% variance of acceptable fan speeds.
warning(4) -	present and operating between 20 to 40% variance of acceptable fan speeds.

Sequence — `::= { fcswChasFanEntry 4 }`

---

fcswChasFanValue

Syntax — INTEGER (0 .. 65535)

Max-Access — read-only

Status — current

Description — The current value (in RPMS) of the measured FAN

Sequence — `::= { fcswChasFanEntry 5 }`

Power Supply

fcswChasPowerSupply OBJECT IDENTIFIER ::= { nishanFCswChassis 11 }

---

fcswChasPowerSupplyTable

Syntax — SEQUENCE OF FcswChasPowerSupplyEntry

Max-Access — not-accessible

Status — current

Description — This table contains information about power supplies.

Sequence — `::= { fcswChasPowerSupply 1 }`

---

fcswChasPowerSupplyEntry

Syntax — FcswChasPowerSupplyEntry

Max-Access — not-accessible

Status — current

Description	Contains information regarding a power supply unit.
	INDEX { fcswChasPowerSupplyId }
Sequence	`::= { fcswChasPowerSupplyTable 1 }`
	FcswChasPowerSupplyEntry ::=
	SEQUENCE {
	fcswChasPowerSupplyId INTEGER,
	fcswChasPowerSupplyOperStatus INTEGER,
	fcswChasPowerSupplyPartNumber DisplayString }

---

fcswChasPowerSupplyId

Syntax	INTEGER(1..2)
Max-Access	read-only
Status	current
Description	Identifier for a power supply instance.
Sequence	`::= { fcswChasPowerSupplyEntry 1 }`

---

fcswChasPowerSupplyOperStatus

Syntax	INTEGER {unknown(1), empty(2), up(3), down(4) }
Max-Access	read-only
Status	current
Description	Actual status of the power supply:
	unknown(1) - status can not be determined.
	empty(2) - power supply not installed.
	up(3) - present and supplying power.
	down(4) - present, but failure indicated.
Sequence	`::= { fcswChasPowerSupplyEntry 2 }`

---

fcswChasPowerSupplyPartNumber

Syntax	DisplayString (SIZE (0..6))
Max-Access	read-only

Status current

Description Power Supply Part Number.

Sequence `::= { fcswChasPowerSupplyEntry 3 }`

Expansion Card Table

fcswCard OBJECT IDENTIFIER ::= { nishanFCswChassis 12 }

fcswCardTable

Syntax SEQUENCE OF FcswCardEntry

Max-Access not-accessible

Status current

Description A table that contains information about the expansion cards in this chassis. For those slots that do not contain a physical card, the table may be implemented to contain a conceptual row with the type of physical module set to 'chasCardEmpty', or it may be implemented to have no conceptual row instance.

Sequence `::= { fcswCard 1 }`

fcswCardEntry

Syntax FcswCardEntry

Max-Access not-accessible

Status current

Description Card attributes

INDEX { fcswCardIndex }

Sequence `::= { fcswCardTable 1 }`

FcswCardEntry ::=

SEQUENCE {

fcswCardIndex	INTEGER,
fcswCardType	INTEGER,
fcswCardSerialNumber	DisplayString,

fcswCardHardwareRevision	DisplayString,
fcswCardOperStatus	INTEGER,
fcswCardPartNumber	DisplayString }

fcswCardIndex

Syntax	INTEGER (1..8)
Max-Access	read-only
Status	current
Description	A unique value for each expansion module within the chassis. This value is determined by the chassis slot number where the module is inserted. Valid entries are 1 to the value of fcswChasNumSlots.
Sequence	`::= { fcswCardEntry 1 }`

fcswCardType

Syntax	INTEGER {	
	other(1),	no card installed
	twoxfcGe(2),	Two dual-mode FC/GE connections
	twoxutpGe(3),	Two UTP GE connections
	twoxetherGbic(4),	Two 70-100km Ethernet GBICs
	twoxetherGbicCams(5),	Two Ethernet GBIC or UTP ports with CAMs
	twoxatm(6),	Two ATM ports
	twoxsonet(7),	Two SONET ports
	twoxgeFc(8),	Two GE/FC ports with Gateway functionality
	routerModule(9),	Intelligent Storage Router Module
	twoxescon(10),	Two ESCON Links
	twoxiFCP(11),	Two iFCP WAN link ports
	twoxutpgbic(12)	Two UTP FastEthernet and two GBIC FC/GE ports }
Max-Access	read-only	
Status	current	
Description	Used to indicate a card type.	

Sequence ::= { fcswCardEntry 2 }

fcswCardSerialNumber

Syntax DisplayString (SIZE (0..8))
Max-Access read-only
Status current
Description Unique serial number for this card.
Sequence ::= { fcswCardEntry 3 }

fcswCardHardwareRevision

Syntax DisplayString (SIZE (0..8))
Max-Access read-only
Status current
Description Hardware Version
Sequence ::= { fcswCardEntry 4 }

fcswCardOperStatus

Syntax INTEGER {up(1), down(2), testing(3), unknown(4) }
Max-Access read-only
Status current
Description Used to indicate the operational status of this card.
Sequence ::= { fcswCardEntry 5 }

fcswCardPartNumber

Syntax DisplayString (SIZE (0..6))
Max-Access read-only
Status current
Description Card Part Number, aka PCA Assembly number.
Sequence ::= { fcswCardEntry 6 }

Port Table

fcswPort OBJECT IDENTIFIER ::= { nishanFCswChassis 13 }

fcswPortTable

Syntax	SEQUENCE OF FcswPortEntry
Max-Access	not-accessible
Status	current
Description	Port table
Sequence	`::= { fcswPort 1 }`

fcswPortEntry

Syntax	FcswPortEntry
Max-Access	not-accessible
Status	current
Description	Entry containing info. for a particular port INDEX { fcswPortIndex }
Sequence	`::= { fcswPortTable 1 }` FcswPortEntry ::= SEQUENCE {

fcswPortIndex	INTEGER,
fcswPortType	INTEGER,
fcswPortConnectorType	INTEGER,
fcswPortAdminStatus	INTEGER,
fcswPortOperStatus	INTEGER,
fcswPortOperation	INTEGER,
fcswPortBeacon	INTEGER,
fcswPortName	OCTET STRING,
fcswPortUtilization	INTEGER,
fcswAutoNegotiationsAdmin	INTEGER,
fcswPortAutoLinkAggregationAdminStatus	INTEGER,

fcswPortConfiguredSpeed	INTEGER,
fcswPortSupportsTCP	INTEGER,
fcswPortActualSpeed	INTEGER }

fcswPortIndex

Syntax INTEGER (1..32)

Max-Access read-only

Status current

Description An index value that uniquely identifies a port. This value is similar to ifIndex in MIB2.

Sequence `::= { fcswPortEntry 1 }`

fcswPortType

Syntax INTEGER {

other(0),	Changed port type from any GIGE types <> any FC types
g1000basefTrunk(1),	Gigabit Ethernet Trunk port
g1000basef(2),	Gigabit Ethernet port
fcAuto(3),	Fibre Channel Auto port
fcFl(4),	Fibre Channel FL port
fcF(5),	Fibre Channel F port
fcIsl(6),	Fibre Channel port for connecting to 3rd party FC switch
g1000basefL3(7),	Gigabit Ethernet L3 Port
g1000basefTrunkCOS(8),	Gigabit Ethernet Trunk port with COS
g1000basefCOS(9),	Gigabit Ethernet port with COS
fcRport(10),	R_Port inter-FC switch connection
g1000iFCP(11),	iFCP WAN Link
g1000iSCSI(12),	iSCSI
g1000TCP(13),	iFCP and iSCSI
g1000iFCPL3(14),	iFCP L3 port
g1000iSCSIL3(15),	iSCSI L3 Port

g1000TCPL3(16),	iFCP and iSCSI L3 port
fcL(17),	Fibre Channel private loop
18	reserved for future use
19	reserved for future use
20	reserved for future use
a100basetTrunk(21),	Fast Ethernet Trunk port
a100baset(22),	Fast Ethernet port
a100basetL3(23),	Fast Ethernet L3 Port
a100basetTrunkCOS(24),	Fast Ethernet Trunk port with COS
a100basetCOS(25),	Fast Ethernet port with COS
a100iFCP(26),	Fast Ethernet with iFCP WAN Link
a100iSCSI(27),	Fast Ethernet with iSCSI
a100TCP(28),	Fast Ethernet with iFCP and iSCSI
a100iFCPL3(29),	Fast Ethernet with iFCP L3 port
a100iSCSIL3(30),	Fast Ethernet with iSCSI L3 Port
a100TCPL3(31)	Fast Ethernet with iFCP and iSCSI L3 }

Max-Access read-write

Status current

Description Port type which will be used after next reset if this variable is set in NVRAM.

NOTE: 1. Ports support either iFCP or iSCSI at one time.

NOTE: 2. Fast Ethernet port types with iSCSI or iFCP are supported only on the IPS 3350. The remaining Fast Ethernet port types will be supported in a future release.

Sequence `::= { fcswPortEntry 2 }`

fcswPortConnectorType

Syntax	INTEGER {	
	empty(0),	no connector detected
	fibreSw(1),	Short-wave transceiver
	fibreLw(2),	Long-wave transceiver
	copperHssdc(3),	High Speed Serial Data Connector
	copperRj45(4),	RJ45 connector
	other(5),	connector type not recognized
	copperHssdc2(6)	High Speed Serial Data Connector 2 }
Max-Access	read-only	
Status	current	
Description	Type of connector installed on this port.	
Sequence	`::= { fcswPortEntry 3 }`	

fcswPortAdminStatus

Syntax	INTEGER { enable(1), disable(2) }
Max-Access	read-write
Status	current
Description	Used to indicate the administrative status of this port.
Sequence	`::= { fcswPortEntry 4 }`

fcswPortOperStatus

Syntax	INTEGER { up(1), down(2), noSignal(3), waitingForReset(4), inTransition(5)}
Max-Access	read-only
Status	current

Description — Indicates the operational status of this port. This is similar to ifOperStatus, but returns more detail:

up(1)	port and link are up, traffic can flow.
down(2)	hardware or driver failure, or port is disabled.
noSignal(3)	hardware is up, but there's no link or no GBIC/SFP.
waitingForReset(4)	port unusable until reset, due to config changes.
inTransition(5)	port is in the process of being enabled or disabled. When a TCP port is manually disabled (by setting fcswPortAdminStatus to 2) it may take the port up to a minute to close iSCSI and iFCP sessions. During this time, fcswPortOperStatus returns inTransition(5). When the port disable is finished, fcswPortOperStatus changes to down(2).

Sequence — `::= { fcswPortEntry 5 }`

---

fcswPortOperation

Syntax — INTEGER { reset(1), lipenable(2), fairenable(3), fairdiable(4) }

Max-Access — read-write

Status — current

Description — " "

Sequence — `::= { fcswPortEntry 6 }`

---

fcswPortBeacon

Syntax — INTEGER { off(0), on(1) }

Max-Access — read-write

Status — current

Description — Turn this on/off to blink the port.

Sequence — `::= { fcswPortEntry 7 }`

fcswPortName

Syntax	OCTET STRING (SIZE (0 .. 32))
Max-Access	read-write
Status	current
Description	A user-defined name for this port.
Sequence	`::= { fcswPortEntry 8 }`

fcswPortUtilization

Syntax	INTEGER (0 .. 2147483647)
Max-Access	read-only
Status	current
Description	The best estimate of the mean physical layer network utilization on this port during the last 5 second sampling interval, in percent. The upper 16-bit word contains the TX utilization, while the lower 16-bit word contains the RX utilization.
Sequence	`::= { fcswPortEntry 9 }`

fcswAutoNegotiationsAdmin

Syntax	INTEGER { notAvailable(0), enable(1), disable(2) }
Max-Access	read-write
Status	current
Description	Used to enable/disable the auto negotiations to any GigE ports. It will be a read-only object if applied to FC ports and return notAvailable.
Sequence	`::= { fcswPortEntry 10 }`

fcswPortAutoLinkAggregationAdminStatus

Syntax	INTEGER { notAvailable(0), enable(1), disable(2) }
Max-Access	read-write
Status	current

Description — Used to enable/disable the auto link aggregation on a GE port. fcswPortAutoLinkAggregationAdminStatus is read-only for FC ports and TCP ports, and always returns notAvailable for those port types.

For software versions or hardware models that do not support Link Aggregation, this variable is read-only and returns notAvailable(0) for all ports. For example, E/OSi version 4.7 does not support link aggregation.

Sequence — `::= { fcswPortEntry 11 }`

---

fcswPortConfiguredSpeed

Syntax — INTEGER (0 .. 10000000)

Max-Access — read-write

Status — current

Description — Used to set the port speed (in Kbits/s) of Ethernet and Fibre Channel ports.

In software versions before 4.7, Ethernet ports support a range from 1500 to 1000000. In software version 4.7 and later, the allowed range for Ethernet ports is 100 to 1000000.

For Fibre Channel ports that support 2Gbps speeds, the valid values are 0 to indicate 'auto' mode, 1000000 for 1Gbps or 2000000 for 2Gbps speeds.

If the Fibre Channel port doesn't support 2Gbps speeds, the speed can only be set to 1Gbps. 2Gbps Fibre Channel is supported only on the Eclipse 2640 and 2600.

Prior to software version 4.0, this variable was called fcswEthernetPortSpeed, and has since been changed to support 2G FC and potential 10G speeds.

Sequence — `::= { fcswPortEntry 12 }`

---

fcswPortSupportsTCP

Syntax — INTEGER { false(0), true(1) }

Max-Access — read-only

Status — current

Description — Indicates whether this port supports TCP, i.e., whether the port type can be set to iFCP or iSCSI. Only certain ports on some SAN Router models support TCP protocols. The value of this variable depends only on the model and port number, not on the current configuration.

Sequence — `::= { fcswPortEntry 13 }`

fcswPortActualSpeed

Syntax — INTEGER (1500 .. 10000000)

Max-Access — read-only

Status — current

Description — The actual speed (in Kbits/s) that the port can deliver, based on fcswPortConfiguredSpeed. For software releases before 4.7, the SAN Router adjusts Ethernet port speeds down to the closest multiple of 1500 Kbps. For software release 4.7 and later, the speed is not rounded down.

In the case of Fibre Channel ports that support 2Gbps speeds, a configured value set to 'auto' will have an actual speed of either 1 Gbps (1000000) or 2 Gbps (2000000).

Sequence — `::= { fcswPortEntry 14 }`

FC Port Table

fcswFCPortTable

Syntax — SEQUENCE OF FcswFCPortEntry

Max-Access — not-accessible

Status — current

Description — FC Port table

Sequence — `::= { fcswPort 2 }`

fcswFCPortEntry

Syntax — FcswFCPortEntry

Max-Access — not-accessible

Status — current

Description	Entry containing info. for a particular port
	INDEX { fcswFCPortIndex }
Sequence	`::= { fcswFCPortTable 1 }`
	FcswFCPortEntry ::= SEQUENCE {
	fcswFCPortIndex INTEGER,
	fcswFCPortRequestNLPortsCur INTEGER,
	fcswFCPortRTTOVcur INTEGER,
	fcswFCPortEDTOVcur INTEGER,
	fcswFCPortRATOVcur INTEGER,
	fcswFCPortLPTOVcur INTEGER,
	fcswFCPortBbCreditCur INTEGER,
	fcswFCPortRequestNLPortsNext INTEGER,
	fcswFCPortRTTOVnext INTEGER,
	fcswFCPortEDTOVnext INTEGER,
	fcswFCPortRATOVnext INTEGER,
	fcswFCPortLPTOVnext INTEGER,
	fcswFCPortBbCredits Integer32 }

fcswFCPortIndex

Syntax	INTEGER (1..32)
Max-Access	read-only
Status	current
Description	An index value that uniquely identifies a port. This value is similar to ifIndex in MIB2.
Sequence	`::= { fcswFCPortEntry 1 }`

fcswFCPortRequestNLPortsCur

Syntax	INTEGER (0..125)
Max-Access	read-only
Status	current
Description	number of FC ports in this FC virtual loop port.

Sequence ::= { fcswFCPortEntry 2 }

fcswFCPortRTTOVcur

Syntax INTEGER (1..4095)
Max-Access read-only
Status current
Description Receiver Transmitter Timeout (in units of thousands of a second).
Sequence ::= { fcswFCPortEntry 3 }

fcswFCPortEDTOVcur

Syntax INTEGER (1..64)
Max-Access read-only
Status current
Description Error Detect Timeout (in seconds).
Sequence ::= { fcswFCPortEntry 4 }

fcswFCPortRATOVcur

Syntax INTEGER (1..4)
Max-Access read-only
Status current
Description Resource Allocation Timeout (in seconds).
Sequence ::= { fcswFCPortEntry 5 }

fcswFCPortLPTOVcur

Syntax INTEGER (1..64)
Max-Access read-only
Status current
Description Loop Timeout (in seconds).
Sequence ::= { fcswFCPortEntry 6 }

fcswFCPortBbCreditCur

Syntax	INTEGER (1..2147483647)
Max-Access	read-only
Status	deprecated
Description	The total number of receive buffers available for holding Class 2 or 3 frames from the attached NxPort. BB Credits are used for flow control of received traffic. This is a read-only duplicate of *fcswFCPortBbCredits*.
Sequence	`::= { fcswFCPortEntry 7 }`

fcswFCPortRequestNLPortsNext

Syntax	INTEGER (0..125)
Max-Access	read-write
Status	current
Description	Number of FC ports in this FC virtual loop port. This value will be used in next FC login. NVRAM value.
Sequence	`::= { fcswFCPortEntry 8 }`

fcswFCPortRTTOVnext

Syntax	INTEGER (1..4095)
Max-Access	read-write
Status	current
Description	Receiver Transmitter Timeout This value will be used in next FC login. NVRAM value. (in units of thousands of a second).
Sequence	`::= { fcswFCPortEntry 9 }`

fcswFCPortEDTOVnext

Syntax	INTEGER (1..64)
Max-Access	read-write
Status	current

Description Error Detect Timeout. This value will be used in next FC login. NVRAM value. (in seconds).

Sequence `::= { fcswFCPortEntry 10 }`

---

fcswFCPortRATOVnext

Syntax INTEGER (1..4)

Max-Access read-write

Status current

Description Resource Allocation Timeout. This value will be used in next FC login. NVRAM value. (in seconds).

Sequence `::= { fcswFCPortEntry 11 }`

---

fcswFCPortLPTOVnext

Syntax INTEGER (1..64)

Max-Access read-write

Status current

Description Loop Timeout. This value will be used in next FC login. NVRAM value. (in seconds).

Sequence `::= { fcswFCPortEntry 12 }`

---

fcswFCPortBbCredits

Syntax Integer32 (1..2147483647)

Max-Access read-write

Status current

Description The total number of receive buffers available for holding Class 2 or 3 frames from the attached NxPort. BB Credits are used for flow control of received traffic. The maximum allowed value depends on the port type, hardware model, and software version. The default value is 16.

Sequence `::= { fcswFCPortEntry 13 }`

FC R_Port Table

This table contains parameters specific to R_Ports. R_Ports are routing ports that connect to other vendor's FC switches using an inter-switch protocol similar to E_Ports.

This table only contains rows for ports currently configured as R_Ports.

fcswEPortTable

Syntax: SEQUENCE OF FcswEPortEntry

Max-Access: not-accessible

Status: current

Description: This table contains port parameters specific to R_Ports. R_Ports are ports that connect to another vendor's FC switches using an inter-switch protocol.

This table contains one row for each port currently configured as an R_Port.

Sequence: `::= { fcswPort 3 }`

fcswEPortEntry

Syntax: FcswEPortEntry

Max-Access: not-accessible

Status: current

Description: Entry containing info. for a particular R_Port.

INDEX { fcswEPortIndex }

Sequence: `::= { fcswEPortTable 1 }`

FcswEPortEntry ::= SEQUENCE {

fcswEPortIndex	INTEGER,
fcswEPortRole	INTEGER,
fcswEPortPreferredDomainID	INTEGER,
fcswEPortCurrentDomainID	INTEGER,
fcswEPortStatus	INTEGER,
fcswEPortPrincipalFabricPortName	WWNtype,

fcswEPortSoIPZoneSetPolicy	INTEGER,
fcswEportMergeFCZone	INTEGER,
fcswEportMergeFCZoneStatus	INTEGER,
fcswEportPortWWN	WWNtype,
fcswEportSwitchInterconnect	INTEGER,
fcswEportAllowFcPortZoning	INTEGER,
fcswEportFcSanId	INTEGER,
fcswEPortZoneSetActionStatus	INTEGER,
fcswEPortSoIPZoneCleanup	INTEGER,
fcswEPortNeighborEDTOV	INTEGER,
fcswEPortNeighborRATOV	INTEGER,
fcswEPortErrorMsgString	OCTET STRING,
fcswEPortInsistentDomainID	INTEGER,
fcswEPortFcNodeWWN	WWNtype }

fcswEPortIndex

Syntax	INTEGER (1..32)
Max-Access	read-only
Status	current
Description	An index value that uniquely identifies a port. The value is the port number, same as fcswPortIndex.
Sequence	`::= { fcswEPortEntry 1 }`

fcswEPortRole

Syntax	INTEGER {principal(1), nonprincipal(2), fabricManager(3) }
Max-Access	read-only
Status	current
Description	This object identifies the role of R_Port.
Sequence	`::= { fcswEPortEntry 2 }`

fcswEPortPreferredDomainID

Syntax	INTEGER(0..255)
Max-Access	read-write
Status	current
Description	DURABLE: { 0xE0:all } This object is used to set the R_Port Preferred Domain ID. A change to this configuration does not take affect until the R_Port is brought down and up.
Sequence	`::= { fcswEPortEntry 3 }`

fcswEPortCurrentDomainID

Syntax	INTEGER(0..255)
Max-Access	read-only
Status	current
Description	This object identifies R_Port Current Domain ID. The current domain ID is 0 for ports that are not R_Ports.
Sequence	`::= { fcswEPortEntry 4 }`

fcswEPortStatus

Syntax INTEGER {

active(1),

isolated(2),

isolatedAtRemoteEnd(3),

error(4),

portNotRport(5),

noSignalDetected(6),

elpParmError(7),

elpParmErrorBPort(8),

elpEDTOVRATOVmismatch(9),

elpClassFmismatch(10),

elpClass3mismatch(11),
establishingLink(12),
reconfigFabric(13),
elpNotReceived(14),
flogiReceived(15),
noMultiSeqBuffer(16),
multiSeqError(17),
sequenceOrder(18),
domainRangeError(19),
loopBack(20),
noResponseELP(21),
noResponseEFP(22),
elpRejected(23),
efpRejected(24),
rdiRejeceted(25),
bfRejeceted(26),
snsAccessError(27),
noAreaId(28),
noResponse(29),
portNotInitialized(30),
activeRoutesToProxiesBlocked(31),
insistentDomainAllocFailure(32),
helloTimeout(33),
proxyDomainInUse(34),
authFailESARejected(35),
authFailEFMDRejected(36),
authFailedSFCRejected(37),
invalidFabricID(38),
authFailedELPResourceFail(39),

authFailedELPPortBindFail(40),

authFailedELPSwitchBindFail(41)

exceededMaxISLPerFabric(42)}

Max-Access read-only

Status current

Description This object identifies the R_Port Error status. See online help in SANvergence for detailed descriptions.

Sequence `::= { fcswEPortEntry 5 }`

---

fcswEPortPrincipalFabricPortName

Syntax WWNtype

Max-Access read-only

Status current

Description This object identifies the Fabric Port Name of the principal R_Port for the given FC SAN Island.

Sequence `::= { fcswEPortEntry 6 }`

NOTE: This variable was called fcswEPortMakeSoIPActive in releases prior to 4.0

---

fcswEPortSoIPZoneSetPolicy

Syntax INTEGER { makeSoIPActiveZoneSet(1), leaveCurrentZoneSet(2), appendSoIPZonesToFcActive(3) }

Max-Access read-write

Status current

Description DURABLE: { leaveCurrentZoneSet:all }

This object determines the action that is taken by Nishan E-Port with the connected FC SAN island. If set to *MakeSoIPActiveZoneSet*, this switch will always force the *SoIPZoneset* on a given island to be the active configuration, otherwise if set to *leaveCurrentZoneset* the current active ZoneSet is not altered. If set to *appendSoIPZonesToFcActive*, the SAN Router zones shall be appended

to the active FC zoneset on the FC Island. If there is no active FC zoneset, no append shall occur.

These actions apply only if the given R_Port is Principal.

NOTE: This variable was called fcswEPortMakeSoIPActive in releases prior to 4.0.

Sequence `::= { fcswEPortEntry 7 }`

fcswEportMergeFCZone

Syntax INTEGER { performMergeNow(1), noMerge(2) }

Max-Access read-write

Status current

Description This object causes the active zones on an FC island to be added into the SAN Router zone configuration. All zones added to the SAN Router side will be preceded with FCSW<PORTID>. The resulting name will be FCSW<PORTID>_<original zone name>.

PORTID is defined to be the last three bytes of the MAC address for the given R_Port. Note merge only occurs for WWN zoning. Domain, port zoning is not merged into the SAN Router zone configuration.

Sequence `::= { fcswEPortEntry 8 }`

fcswEportMergeFCZoneStatus

Syntax INTEGER { mergeCompleted(1), mergeFailed(2), mergeInProgress(3), noMerge(4) }

Max-Access read-only

Status current

Description This objects shows whether a merge with the FC island has been completed or not.

Sequence `::= { fcswEPortEntry 9 }`

fcswEportPortWWN

Syntax WWNtype

Max-Access	read-only
Status	current
Description	This objects is the port WWN for the given R_Port.
Sequence	`::= { fcswEPortEntry 10 }`

fcswEportSwitchInterconnect

Syntax	INTEGER { brocade(1), interoperable(2), mcdata(3) }
Max-Access	read-write
Status	current
Description	DURABLE: {1:all} This object shows what interoperability mode an R_Port operates in. Note if running in interOperable mode Domain IDs are limited to a value between hex 61..7F. InterOperabilty mode defines a mode used by McData, and Brocade switches to interconnect to one another. McData mode requires the Domain IDs to be a value between hex 1..1F. A change to this configuration does not take affect until the R_Port is brought down and up. DEFVAL {interoperable}
Sequence	`::= { fcswEPortEntry 11 }`

fcswEportAllowFcPortZoning

Syntax	INTEGER { allowed(1), disallowed(2) }
Max-Access	read-write
Status	current
Description	DURABLE: {1:all} This object informs R_Port to allow or disallow port zoning on a given FC island. A change to this configuration does not take affect until the R_Port is brought down and up. DEFVAL {allowed}
Sequence	`::= { fcswEPortEntry 12 }`

fcswEportFcSanId

Syntax	INTEGER (1..85)
Max-Access	read-write
Status	current
Description	DURABLE: {1:all} This object determines the Fibre Channel SAN ID to be used on this eport. All eports on the same FC san must have the same Fibre Channel SAN ID. A change to this configuration does not take affect until the R_Port is brought down and up. DEFVAL {1}
Sequence	`::= { fcswEPortEntry 13 }`

NOTE: Following e-port variables are unsupported in 4.0 release and before

fcswEPortSoIPZoneCleanup

Syntax	INTEGER { noCleanUp(1), removeIpsZoning(2) }
Max-Access	read-write
Status	current
Description	This MIB object is valid only when the fcswEPortSoIPZoneSetPolicy is set to leaveCurrentZoneset(2). Setting this variable to removeIpsZoning(2) will remove all IPS Zone Sets, and associated Zones, from the connected FC fabric. If an IPS Zone Set was the active Zone Set, there will no longer be an active FC Zone Set.
Sequence	`::= {fcswEPortEntry 14}`

fcswEPortZoneSetActionStatus

Syntax	INTEGER (0..2147483647)
Max-Access	read-only
Status	current
Description	This bitmap identifies the status after a SAN Router Zone action is taken. Following bits represent the status codes which are returned. Bit 0 is the least significant bit.

Bits 0-15 indicate the status of the action.

Bits 16-31 indicate the information about the configuration.

Bit	Status
0	Configuration matches or appended successfully.
1	Configuration update in progress.
2	Configuration mismatches.
3	Append failed. No active zoneset.
4	Append zone removal failed. Last zones in active zoneset.
5	Zoning Error Detected. See fcswEPortErrorMsgString for details.
16	Configuration mismatches due to 'Leave Current Zoneset' policy.
17	Cleanup failed. Not in 'Leave Current Zoneset' policy.
18	No active FC Zoneset on the FC Switch.
19	SAN Router Zoneset is the active zoneset.
20	FC Fabric Defined Zoneset is the active zoneset.

Bits 0-3 are transitional states while applying a policy for this R_Port.

Bit 16 is set when our R_Port is set to 'Leave Current Zoneset' policy and the connected FC neighbor has another active FC zoneset.

Bit 17 is set when the user manually tries to set *fcswEPortSoIPZoneCleanup* variable from the mib when the policy is not set to 'Leave Current Zoneset'.

Bits 18-20 are useful indications of the configuration on the connected FC switch on this R_Port. All other bits are reserved.

Sequence `::= { fcswEPortEntry 15 }`

---

fcswEPortNeighborEDTOV

Syntax INTEGER (1..2147483)

Max-Access read-only

Status current

Description The E_D_TOV of the connected neighbor in seconds. Only valid when port is connected. The FC-PH standard defines this value in ms, but to be consistent across our mib, we show it in secs.

Sequence ::= { fcswEPortEntry 16 }

fcswEPortNeighborRATOV

Syntax INTEGER (1..2147483)

Max-Access read-only

Status current

Description The R_A_TOV of the connected neighbor in seconds. Only valid when port is connected. The FC-PH standard defines this value in ms, but to be consistent across our mib, we show it in secs.

Sequence ::= { fcswEPortEntry 17 }

fcswEPortErrorMsgString

Syntax OCTET STRING(SIZE(0..255))

Max-Access read-only

Status current

Description This object is used by R_Port to convey any more detailed error information for last error detected for a given R_Port.

Sequence ::= { fcswEPortEntry 18 }

fcswEPortInsistentDomainID

Syntax INTEGER { enabled(1), disabled(2) }

Max-Access read-write

Status current

Description DURABLE: { disabled:all }

This object is used by R_Port to determine whether the preferredDomainID is insistent or not. If this is enabled, and the R_Port cannot acquire the *preferredDomainID* this R_Port will isolate.

NOTE: If enterpriseFabricMode is enabled, a set to disable InsistentDomainID will result in an error.

A change to this configuration does not take affect until the R_Port is brought down and up.

Sequence ::= { fcswEPortEntry 19 }

fcswEPortFcNodeWWN

Syntax WWNtype

Max-Access read-only

Status current

Description This objects represents this port's proxy FC Node WWN on the FC Fabric. The 0x7E, and 0x7F proxy WWN can be found in the eportSubFabricTable as eportSubFabricFcFabricProxyWWN, and eportSubFabricLocalProxyWWN.

Sequence ::= { fcswEPortEntry 20 }

TCP Port Table

This contains port parameters specific to TCP ports. TCP ports support iFCP or iSCSI or both. This table contains one row for each port currently configured as a TCP port.

fcswTCPPortTable

Syntax SEQUENCE OF FcswTCPPortEntry

Max-Access not-accessible

Status current

Description Contains port parameters specific to TCP ports. TCP ports support iFCP or iSCSI or both. This table contains one row for each port currently configured as a TCP port.

Sequence ::= { fcswPort 4 }

fcswTCPPortEntry

Syntax FcswTCPPortEntry

Max-Access not-accessible

Status current

Description Entry containing info for a particular TCP port.

INDEX { fcswTCPPortIndex }

Sequence `::= { fcswTCPPortTable 1 }`

```
FcswTCPPortEntry ::= SEQUENCE {
fcswTCPPortIndex                                INTEGER,
fcswTCPIpAddressCurrent                         IpAddress,
fcswTCPSubnetMaskCurrent                        IpAddress,
fcswTCPDefaultGatewayAddressCurrent             IpAddress,
fcswTCPProxyIpAddressCurrent                    IpAddress,
fcswTCPIpAddressOnNextReset                     IpAddress,
fcswTCPSubnetMaskOnNextReset                    IpAddress,
fcswTCPDefaultGatewayAddressOnNextReset IpAddress,
fcswTCPProxyIpAddressOnNextReset                IpAddress,
fcswTCPAutoReset                                INTEGER,
fcswTCPResetNow                                 TruthValue,
fcswTCPOptions                                  INTEGER,
fcswTCPMaxMtuSize                               INTEGER,
fcswTCPiSCSIOptions                             INTEGER,
fcswTCPiSCSIFirstBurstLength                    INTEGER,
fcswTCPiSCSIMaxBurstLength                      INTEGER,
fcswTCPiSCSIMaxRecvDataSegmentLength            INTEGER,
fcswTCPiSNSServerIpAddress                      IpAddress,
fcswTCPCompressionMethod                        INTEGER,
fcswTCPiSCSITargetChapCredential                OCTET STRING,
fcswTCPiSCSILoginRetryTimeout                   INTEGER,
fcswTCPReorderThreshold                         Integer32,
fcswTCPFastWriteOptions                         INTEGER,
fcswTCPFastWriteMaxXferRdys                     INTEGER,
fcswTCPFastWriteBufferMem                       INTEGER,
fcswTCPXmitBufferMgmtMem                        INTEGER,
fcswTCPXmitMgmtMaxBufferCount                   INTEGER,
fcswTCPXmitMgmtCurrentUsedBufferCount           INTEGER }
```

fcswTCPPortIndex

Syntax	INTEGER (1..32)
Max-Access	read-only
Status	current
Description	An index value that uniquely identifies a port. The value is the port number, same as *fcswPortIndex*.
Sequence	`::= { fcswTCPPortEntry 1 }`

fcswTCPIpAddressCurrent

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The IP Address currently in use for this port if the port type is set to iFCP or iSCSI or both. For other port types the value of this variable is undefined. For layer 3 UDP ports, the port's current IP address is specified in the NISHAN-RPRO MIB.
Sequence	`::= {fcswTCPPortEntry 2}`

fcswTCPSubnetMaskCurrent

Syntax	IpAddress
Max-Access	read-only
Status	current
Description	The subnet mask currently in use for this port if the port type is set to iFCP or iSCSI or both. For other port types the value of this variable is undefined. For layer 3 UDP ports, the port's current subnet mask is specified in the NISHAN-RPRO MIB.
Sequence	`::= {fcswTCPPortEntry 3}`

fcswTCPDefaultGatewayAddressCurrent

Syntax	IpAddress
Max-Access	read-write

Status current

Description The default gateway IP address currently in use for this port if the port type is set to iFCP or iSCSI or both. For other port types the value of this variable is undefined. For layer 3 UDP ports, the port's default gateway is specified in the NISHAN-RPRO MIB.

TCP ports act as end nodes (hosts) attached to the WAN, and therefore may have a different default gateway than the UDP ports connecting to the local SAN.

Sequence `::= {fcswTCPPortEntry 4}`

---

fcswTCPProxyIpAddressCurrent

Syntax IpAddress

Max-Access read-only

Status current

Description The TCP proxy IP address currently in use for this port if the port type is set to iFCP or iSCSI or both. For other port types the value of this variable is undefined.

The SAN Router represents remote iFCP devices and iSCSI devices on the local SAN by proxy. The proxy IP address is registered for these devices in the Storage Name Service. The SAN Router performs address translation when forwarding traffic from the local UDP SAN to the TCP WAN.

NOTE: The proxy IP address must be in the same IP subnet as the switch address. The proxy IP address cannot be shared with other TCP ports or other devices.

Sequence `::= {fcswTCPPortEntry 5}`

---

fcswTCPIpAddressOnNextReset

Syntax IpAddress

Max-Access read-write

Status current

Description DURABLE: { 0:all }

The IP Address to use for this port after the next system reset if the port type is iFCP or iSCSI or both. If the port type is GE Layer 3, the port's IP address is specified in the NISHAN-RPRO MIB instead.

Sequence `::= {fcswTCPPortEntry 6}`

---

fcswTCPSubnetMaskOnNextReset

Syntax IpAddress

Max-Access read-write

Status current

Description DURABLE: { 0:all }

The subnet mask to use for this port after the next system reset if the port type is iFCP or iSCSI or both. If the port type is GE Layer 3, the port's subnet mask is specified in the NISHAN-RPRO MIB instead.

Sequence `::= {fcswTCPPortEntry 7}`

---

fcswTCPDefaultGatewayAddressOnNextReset

Syntax IpAddress

Max-Access read-write

Status current

Description DURABLE: { 0:all }

The default gateway address to use for this port after the next system reset if the port type is iFCP or iSCSI or both. TCP ports act as end nodes (hosts) attached to the WAN, and therefore may have a different default gateway than the UDP ports connecting to the local SAN.

Sequence `::= {fcswTCPPortEntry 8}`

---

fcswTCPProxyIpAddressOnNextReset

Syntax IpAddress

Max-Access read-write

Status current

Description DURABLE: { 0:all }

The TCP proxy IP address to use for this port after the next reset if the port type is iFCP or iSCSI or both. The SAN Router represents remote iFCP devices and iSCSI devices on the local SAN by proxy. The proxy IP address is registered for these devices in the Storage Name Service. The SAN Router performs address translation when forwarding traffic from the local UDP SAN to the TCP WAN.

NOTE: The proxy IP address must be in the same IP subnet as the switch address. The proxy IP address cannot be shared with other TCP ports or other devices.

Sequence `::= {fcswTCPPortEntry 9}`

fcswTCPAutoReset

Syntax INTEGER { enable(1), disable(2) }

Max-Access read-write

Status current

Description DURABLE: { 1:all }

If AutoReset is enabled, the TCP port resets itself after a fatal error. If AutoReset is disabled, errors do not force an automatic reset.

DEFVAL {enable}

Sequence `::= {fcswTCPPortEntry 10}`

fcswTCPResetNow

Syntax TruthValue

Max-Access read-write

Status current

Description Setting this variable to true(1) causes the TCP port to reset itself immediately. iSCSI and iFCP traffic is interrupted. The TCP port configuration is updated to the latest configuration settings. Reading this variable returns true(1) if a port reset is in progress, or false(2) if a port reset is finished or no port reset has been requested.

Sequence `::= {fcswTCPPortEntry 11}`

fcswTCPOptions

Syntax	INTEGER (0..2147483647)
Max-Access	read-write
Status	current
Description	DURABLE: { 0:all }

A bitmap representing TCP options for this port. Bit 0 is the least significant bit. Setting a bit enables the option. Current bit assignments are:

bit 0	Manual Fast Write
bit 1	Auto Fast Write
bit 2	Enable SW Compression
bit 3	Automatic SW Compression
bit 4	End-To-End Flow Control
bit 5	Internal Use
bit 6	Use Manually Configured MTU Size
bit 7	Use Min. of Discovered MTU or Manual MTU Sizes
bit 8	Enable Zone based Scheduling (stream based if clear)
bit 9	Enable iSNS Server
bit 10	Enable iSNS Client
bit 11	Enable SACK (Selective ACK) for iSCSI TCP connections
bit 12	Enable SACK (Selective ACK) for iFCP TCP connections
bit 13	Enable Galileo (Network Simulation) Mode for the port
bits 14-31	Reserved
bit 14	Enable Reorder Resistance (allow more duplicate ACKs)
bit 15	Enable Smaller CWND Reduction in Fast Recovery
bit 16	Enable Quick Start (larger initial CWND)
bit 17	Disable Congestion Avoidance
bit 18	Reduced Slow Start Timeout

bit 19	Enable Hardware Compression
bit 20	Enable Transmit Buffer Management
bits 21-31	Reserved

Manual Fast Write(0) enables Fast Write, also known as WAN Optimization, to improve TCP write performance on long-distance iFCP connections.

Auto Fast Write(1) adjusts the TCP performance based on network conditions. Bits 0 & 1 cannot both be set. Auto Fast Write is reserved and may be supported in a future release.

Enable SW Compression(2) causes all data transmitted out this port to be compressed in firmware. Automatic SW Compression(3) causes transmitted data to be compressed only when necessary, depending on traffic rate and remote link bandwidth. If neither option is enabled, transmitted data is not compressed, but compressed data may still be received.

If either option is enabled, the MIB variable fcswTCPCompressionMethod specifies the compression algorithm. Bits 2, 3, and 19 are mutually exclusive: at most one may be set at a time. Attempting to set more than one compression mode at a time results in an SNMP SET failure.

Use Manually Configured MTU (6) - configures all TCP connections to use the manually configured MTU, instead of the MTU size discovered for the connection.

Use Min of Discovered MTU or Manual MTU Sizes (7)- configures TCP connections to use either the discovered MTU or the manually configured MTU for the port, whichever is smaller.

If bits 6 & 7 are 0, then auto discovery of the MTU size is used for communication (which is the default behavior). Bits 6 & 7 cannot be set at once.

A TCP port can be enabled to be an iSNS Server or an iSNS Client but not both. Setting bit 9 to 1 will imply that this TCP port is to be configured as an iSNS server. Setting bit 10 to 1 will imply that this TCP port is to be configured as an iSNS client.

Bits 9 and 10 should not be set at once. SACK (Selective ACK) allows acknowledgement of non-contiguous sequence numbers to reduce the amount of retransmitted data when packets are lost.

Set bit 11 to '1' to enable SACK for iSCSI TCP connections on this port. Set bit 11 to '0' to disable SACK for iSCSI TCP connections on this port.

Set bit 12 to '1' to enable SACK for iFCP TCP connections on this port. Set bit 12 to '0' to disable SACK for iFCP TCP connections on this port.

SACK is enabled or disabled for both transmit and receive together. When SACK is enabled, the TCP port accepts SACKs from remote receivers, and may send SACKs to remote transmitters that accept SACKs. When SACK is disabled, the port does not include the Sack-Permitted option in data sent to a remote receiver, and does not send SACKs to remote transmitters even if the remote transmitter accepts SACKs.

When SACK is enabled for a port, individual TCP sessions may or may not use SACK depending on whether the remote end supports SACK. To check whether a particular TCP session is using SACK, see the fcswTCPSessionStatsTable.

Galileo Mode (bit 13) runs the port in a network delay and drop simulation mode. All normal functions of the port are disabled.

Reorder Resistance (bit 14) allows more duplicate ACKs before triggering fast retransmit or fast recovery. The number of duplicates is specified in fcswTCPReorderThreshold. This can improve performance in networks where re-ordered packets occur more often, by reducing unnecessary fast retransmits and fast recoveries due to reordered packets. In some cases Reorder Resistance may delay recovery from dropped packets.

Smaller CWND Reduction (bit 15) reduces the CWND size by 1/8 instead of 1/2 when Fast Recovery occurs. The SAN Router effectively adjusts more slowly to congestion events.

Quick Start (bit 16) uses a larger initial value for the congestion window at the beginning of slow starts, and increases the window size more quickly. This option may be used on dedicated long latency links to improve performance by skipping unnecessary protection against interfering with other traffic.

Congestion Avoidance (bit 17) may be disabled by setting bit 17. Congestion Avoidance is the slow growth mode of the SAN Router's CWND size after a congestion event. Disabling Congestion Avoidance causes the CWND size to increase more rapidly after a congestion event.

Reduced Slow Start Timeout (bit 18), when enabled, reduces the minimum Slow Start timeout from 500 msec to 150 msec.

Hardware Compression (bit 19) enables hardware-based compression if available. Bit 19 has no effect if hardware compression is not available in this hardware model or software bundle. Bits 2, 3, and 19 are mutually exclusive: at most one may be set at a time. Attempting to set more than one compression mode at a time results in an SNMP SET failure.

If bit 20 is set, Transmit Buffer Management is enabled, otherwise disabled.

Reserved bits must be zero when read, and must be ignored by the agent when set, for future backward compatibility.

DEFVAL { 0 }

Sequence `::= {fcswTCPPortEntry 12}`

fcswTCPMaxMtuSize

Syntax INTEGER (512 .. 4096)

Max-Access read-write

Status current

Description The maximum value, in bytes, that can be used for the Maximum Transmission Unit (MTU) Size for this TCP Port. This variable can be used to prevent accidental fragmentation, when the Path (discovered) MTU value, as returned by paths from/to this port, is incorrect. This size is used only if fcswTCPOptions bits Use Manually Configured MTU (6) or Use Min of Discovered MTU or Manual MTU sizes (7) are set to 1 and is ignored if auto discovery of MTU sizes is enabled.

DEFVAL { 1500 }

Sequence `::= { fcswTCPPortEntry 13 }`

fcswTCPiSCSIOptions

Syntax INTEGER (0..2147483647)

Max-Access read-write

Status current

Description DURABLE: { 300:all }

A bitmap representing iSCSI options for this port. This value is used only if this port is configured for iSCSI. Bit 0 is the least significant bit. Setting a bit enables the option. Current bit assignments are:

Bits	Attribute	Default
bit 0:	Reserved	OFF
bit 1:	Digest	OFF
bit 2:	InitialR2T	ON
bit 3:	ImmediateData	ON
bit 4:	StoreAndForward	OFF
bit 5:	Big PDU	ON
bit 6:	READ Padding	OFF
bit 7:	WRITE Padding	OFF
bit 8:	Target NOP	ON
bit 9:	CHAP Authentication	OFF
bit 10:	CHAP Required	OFF
bits 11-31:	Reserved	OFF

Reserved bits must be zero when read, and must be ignored by the agent when set, for future backward compatibility.

Bit 0: Reserved (default OFF)

Not used.

Bit 1: Digest (default OFF)

The initiator/target will negotiate if data integrity is required during Login. Digests enable the checking of end-to-end, non-cryptographic data integrity beyond the integrity checks provided by the link layers and the covering of the whole communication path including all elements that may change the network level PDUs such as routers, switches, and proxies.

Bit 2: InitialR2T (default ON)

The InitialR2T key is used to turn off the default use of R2T for unidirectional and the output part of bidirectional commands thus allowing an initiator to start sending data to a target as if it has received an initial R2T with Buffer Offset=Immediate Data Length

and Desired Data Transfer Length=(min(FirstBurstLength, Expected Data Transfer Length) - Received Immediate Data Length).

Bit 3: ImmediateData (default ON)

The initiator and target negotiate support for immediate data. To turn immediate data off, the initiator or target must state its desire to do so. ImmediateData can be turned on if both the initiator and target have ImmediateData=Yes.

Bit 4: Store and Forward (default OFF)

The target will wait until all the data received from the drive then deliver it to the initiator.

Bit 5: Big PDU (default ON)

The target will send data Header for each data frame.

Bit 6: Read Padding (default OFF)

The target will send the padding bytes to the initiator if the initiator can not handle the underrun data.

Bit 7: Write Padding (default OFF)

The target will send the padding bytes to the drive if the initiator do not pad the underrun data.

Bit 8: Target nop (default ON)

The initiator/target will turn on/off sending the NOP packet.

Bit 9: CHAP Authentication (default OFF)

An iSCSI initiator and target negotiate an authentication method during login. If bit 9 is off, this port does not support CHAP authentication. If bit 9 is on, this port supports CHAP authentication of an attached iSCSI initiator when the port is acting as a virtual iSCSI target.

Bit 10: CHAP Required (default OFF)

If CHAP Authentication is disabled (bit 9 is off) then bit 10 has no effect. If CHAP Authentication is enabled (bit 9 is on), and bit 10 is off, then this port will use CHAP to authenticate initiators that support CHAP, but will also accept logins from initiators that don't support CHAP.

If both bits 9 and 10 are on, then this port only accepts logins from initiators that support CHAP and are successfully authenticated via CHAP.

DEFVAL { 300 }

Sequence `::= { fcswTCPPortEntry 14 }`

---

fcswTCPiSCSIFirstBurstLength

Syntax INTEGER (8..256)

Max-Access read-write

Status current

Description DURABLE: { 64:all }

The maximum amount of unsolicited data an iSCSI initiator may send to the target during the execution of a single SCSI command, kilobytes. The default value is 64. Any value between 8 and 256 is valid. Typical other values are 8, 128 and 256.

This value is used only if this port is configured for iSCSI.

DEFVAL { 64 }

Sequence `::= { fcswTCPPortEntry 15 }`

---

fcswTCPiSCSIMaxBurstLength

Syntax INTEGER (8..256)

Max-Access read-write

Status current

Description DURABLE: { 256:all }

The maximum amount of unsolicited data an iSCSI initiator may send to the target during the execution of a single SCSI command, kilobytes. The default value is 64. Any value between 8 and 256 is valid. Typical other values are 8, 128 and 256. This value is used only if this port is configured for iSCSI.

DEFVAL { 64 }

Sequence `::= { fcswTCPPortEntry 15 }`

---

fcswTCPiSCSIMaxBurstLength

Syntax INTEGER (8..256)

Max-Access read-write

Status current

Description DURABLE: { 256:all }

The maximum SCSI data payload in a Data-In or a solicited Data-Out iSCSI sequence, in kilobytes. The default value is 256. Any value between 8 and 256 is valid. Typical other values are 8, 64, and 128.

This value is used only if this port is configured for iSCSI.

DEFVAL { 256 }

Sequence `::= { fcswTCPPortEntry 16 }`

---

fcswTCPiSCSIMaxRecvDataSegmentLength

Syntax INTEGER (8..256)

Max-Access read-write

Status current

Description DURABLE: { 64:all }

The maximum data segment length the SAN Router can receive in an iSCSI PDU, in kilobytes. The default value is 64. Any value between 8 and 256 is valid. Typical other values are 8, 128 and 256. This value is used only if this port is configured for iSCSI.

DEFVAL { 64 }

Sequence `::= { fcswTCPPortEntry 17 }`

---

fcswTCPiSNSServerIpAddress

Syntax IpAddress

Max-Access read-write

Status current

Description DURABLE: { 0:all }

The IP Address of the iSNS server to which this TCP port registers as an iSNS client.

This value is valid only if this port has been configured as an iSNS client by setting bit 10 of the fcswTcpOptions.

Sequence `::= {fcswTCPPortEntry 18}`

fcswTCPCompressionMethod

Syntax INTEGER {

lzo(1),

lzoWithHistory(2),

lzoWithHistoryAndHuffman(3),

deflate(4),

lzo2ByteWithHistory(5) }

Max-Access read-write

Status current

Description DURABLE: { 1 }

This MIB variable defines the compression algorithm for iFCP connections on this port. This setting is ignored when compression is disabled in fcswTCPPortOptions.

lzo(1) performs compression on a frame-by-frame basis. This method is best when there are many active iFCP initiator-target sessions.

lzoWithHistory(2) performs LZO compression with 8KB of history. It provides a higher compression ratio at the expense of memory. It works best with few active iFCP initiator-target sessions, and a T3 or faster remote link.

lzoWithHistoryAndHuffman(3) performs Huffman encoding after the LZO compression with 8KB of history. It may provide a higher compression ratio at the expense of CPU and memory resources. It also works best with few active iFCP initiator-target sessions, and a medium-speed remote link.

deflate(4) provides maximum compression, but at the lowest rate. It is best for T1 links. The number of active sessions does not matter.

lzo2ByteWithHistory(5) performs LZO compression with history 2 bytes at a time instead of 1 byte at a time. Compared to lzoWithHistory, lzo2ByteWithHistory increases the compression rate at the cost of a lower compression ratio. It works best with few active iFCP initiator- target sessions, and a fast remote link.

DEFVAL {lzo}

Sequence `::= { fcswTCPPortEntry 19 }`

fcswTCPiSCSITargetChapCredential

Syntax	OCTET STRING (SIZE (0 .. 144))
Max-Access	read-write
Status	current
Description	DURABLE: The secret key, in encrypted form, for iSCSI CHAP authentication of this port's virtual iSCSI initiator by iSCSI targets. If this port receives a CHAP challenge from an iSCSI target, this mib variable specifies the secret key to use in preparing the CHAP response. For transfer between the management station and switch, the CHAP secret key is encrypted according to the method described for encrypting user passwords in RFC 2865 section 5.2. The first 16 bytes of the MIB variable value contain the 16-byte random number ('Request Authenticator' in RFC 2865) used as part of the encryption process. The remaining 1 - 128 bytes contain the encrypted CHAP secret key, truncated to the same length as the unencrypted key. The unencrypted CHAP secret key should be at least 16 bytes long for best security. This variable is not implemented in software versions that do not support FC initiators communicating to iSCSI targets.
Sequence	`::= { fcswTCPPortEntry 20 }`

fcswTCPiSCSILoginRetryTimeout

Syntax	INTEGER (0 | 30 .. 600)
Max-Access	read-write
Status	current
Description	DURABLE: { 60:all } When the target is unzoned or disconnected, the iSCSI initiator attempts to relogin to the target. While the iSCSI initiator can continue to relogin to the target indefinitely, there can be a performance issue with the constant retries. This managed object specifies the timeout period, in seconds, in which the iSCSI initiator can attempt to relogin to the target before a status is sent by the switch to the iSCSI initiator to abort the login. After that point, the iSCSI initiator must rediscover the target.

This managed object applies to all the iSCSI sessions bound to the fcswTCPPortIndex. If this manage object is set to 0, there is no limit to the number of login retries. The default value is 60 seconds.

DEFVAL { 60 }

Sequence `::= { fcswTCPPortEntry 21 }`

fcswTCPReorderThreshold

Syntax Integer32 (8..128)

Max-Access read-write

Status current

Description DURABLE: { 8:all }

When Reorder Resistance is enabled in fcswTCPOptions, fcswTCPReorderThreshold specifies the number of duplicate ACKs that may be received within a short time (approximately 2 msec) before a fast retransmit or fast recovery is triggered.

Reorder Resistance reduces unnecessary fast retransmits and fast recoveries due to reordered packets. In some cases Reorder Resistance may delay recovery from dropped packets.

If Reorder Resistance is not enabled, the fcswTCPReorderThreshold value is not used. The default value is 8.

DEFVAL { 8 }

Sequence `::= { fcswTCPPortEntry 22 }`

fcswTCPFastWriteOptions

Syntax INTEGER (0..7)

Max-Access read-write

Status current

Description DURABLE: { 0:all }

A bitmap representing Fast Write options for this port. These options are only meaningful when Fast Write is enabled in fcswTCPOptions.

FastWrite, also known as WAN Optimization, improves TCP write performance on long-distance iFCP connections. The Fast Write options modify the Fast Write behavior. Setting a bit enables the

option. The default values for all options are 'off'. Current bit assignments are:

Bit 0: XFER_RDY Windowing. This option controls the way in which the outstanding transfer ready count is used. If bit 0 is off then then the XFER_RDYs are limited to the value specified in fcswTCPFastWriteMaxXferRdys. If bit 0 is set then the limit in fcswTCPFastWriteMaxXferRdys represents a 'window' of xfer_rdys that can be issued.

Bit 1: Hold Commands. When bit 1 is set, the SAN Router may delay IO commands for some amount of time before sending the command to the target.

Bit 2: Remote Buffering Management. If bit 2 is off then there are no specific limits on the remote buffering resources used by fast write. If bit 2 is set then the initiator side will limit the fast write data in the pipeline according to the amount of buffer memory available, as specified in fcswTCPFastWriteBufferMem.

Bits 3 - 31 are reserved for future use. They are 0 when read, and ignored when written, for future backward compatibility.

DEFVAL { 0 }

Sequence `::= {fcswTCPPortEntry 23}`

---

fcswTCPFastWriteMaxXferRdys

Syntax INTEGER (1 .. 512)

Max-Access read-write

Status current

Description DURABLE: { 512:all }

The maximum number of concurrent outstanding XFER_RDYs allowed per FC Fast Write login session. The limit is applied separately to each session, and each session has the same limit.

DEFVAL { 512 }

Sequence `::= { fcswTCPPortEntry 24 }`

---

fcswTCPFastWriteBufferMem

Syntax INTEGER (1024 .. 65536)

Max-Access read-write

Status current

Description DURABLE: { 32768:all }

The amount of memory, in KBytes, allocated to Fast Write buffers. The memory pool is shared across all FC login sessions for this port. The allowed range corresponds to 1MB to 64MB. The actual value used in the SAN Router is the configured value rounded down to the nearest multiple of 8 KBytes. The default value is 32MB.

DEFVAL { 32768 }

Sequence `::= { fcswTCPPortEntry 25 }`

---

fcswTCPXmitBufferMgmtMem

Syntax INTEGER (256 .. 65536)

Max-Access read-write

Status current

Description DURABLE:

The number of buffers requested by the user for Transmit Buffer Management. This value cannot exceed the value of fcswTCPXmitMgmtMaxBufferCount. The actual number of buffers used by the TCP port is given by fcswTCPXmitMgmtCurrentUsedBufferCount..

DEFVAL { 256 }

Sequence `::= { fcswTCPPortEntry 26 }`

---

fcswTCPXmitMgmtMaxBufferCount

SYNTAX INTEGER (256 .. 44599)

Max-Access read-only

Status current

Description The maximum number of buffers that the user can request for Transmit Buffer Management. This depends on the TCP port speed and is set after port initialization.

DEFVAL { 44599 }

Sequence `::= { fcswTCPPortEntry 27 }`

fcswTCPXmitMgmtCurrentUsedBufferCount

Syntax	INTEGER (256 .. 44599)
Max-Access	read-only
Status	current
Description	The actual number of buffers used by the TCP port for Transmit Buffer Management. DEFVAL { 256 }
Sequence	`::= { fcswTCPPortEntry 28 }`

fcswChasBrdLeds

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	A character string that contains the values of the LEDs for all the ports. Each character represents an LED. It starts with management Port's LEDs info, followed by the LEDs info of port 1, port 2, etc. The total length will be number of ports times number of LEDs per port. If the LED values are not available, a zero length string is returned. The following shows the meaning of a character: value meaning 0 off 1 slow blinking 2 fast blinking 3 on
Sequence	`::= { nishanFCswChassis 14 }`

TCP Port Compression Statistics Table

fcswTcpCompStatTable

Syntax	SEQUENCE OF FcswTcpCompStatEntry
Max-Access	not-accessible

Status current

Description A Table that lists the TCP ports and its compression statistics.
in-compression-ratio = tcpPortDeCompressedIn / tcpPortCompressedIn

out-compression-ratio = tcpPortDeCompressedOut / tcpPortCompressedOut

Sequence `::= { fcswPort 5 }`

---

fcswTcpCompStatEntry

Syntax FcswTcpCompStatEntry

Max-Access not-accessible

Status current

Description Comression statistics entries for the tcp ports.

INDEX {fcswTcpPortIndex}

Sequence `::= { fcswTcpCompStatTable 1 }`

FcswTcpCompStatEntry ::= SEQUENCE {

fcswTcpPortIndex	INTEGER,
fcswTcpPortCompressedIn	Counter32,
fcswTcpPortCompressedOut	Counter32,
fcswTcpPortDeCompressedIn	Counter32,
fcswTcpPortDeCompressedOut	Counter32 }

---

fcswTcpPortIndex

Syntax INTEGER (1..32)

Max-Access read-only

Status current

Description An index value that uniquely identifies a port. The value is the port number.

Sequence `::= { fcswTcpCompStatEntry 1 }`

fcswTcpPortCompressedIn

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total amount of iFCP data received on this TCP Port, before decompression, in units of 16-byte blocks. If compression is set to Auto and the port speed is rate-limited, the received iFCP traffic may be a mix of compressed and uncompressed data. This counter's value is the total received iFCP data, including both the compressed and uncompressed data.
Sequence	`::= { fcswTcpCompStatEntry 2 }`

fcswTcpPortCompressedOut

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total amount of iFCP data transmitted out this TCP port, after compression, in units of 16-byte blocks. If compression is set to Auto and the port speed is rate-limited, the transmitted iFCP traffic may be a mix of compressed and uncompressed data. This counter's value is the total transmitted iFCP data, including both the compressed and uncompressed data.
Sequence	`::= { fcswTcpCompStatEntry 3 }`

fcswTcpPortDeCompressedIn

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total amount of iFCP data received on this TCP Port, after decompression, in units of 16-byte blocks. This is the amount of data forwarded to local non-TCP switch ports, after any compressed data has been decompressed.
Sequence	`::= { fcswTcpCompStatEntry 4 }`

fcswTcpPortDeCompressedOut

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total amount of iFCP data transmitted out this TCP port, before compression, in units of 16-byte blocks. This is the amount of data received from local non-TCP switch.ports to be sent out this TCP port, before the data is compressed.
Sequence	`::= { fcswTcpCompStatEntry 5 }`

TCP Port Storage Statistics Table

This table contains the storage statistics for the TCP ports. For now, only iFCP traffic is included, but eventually iSCSI will be added as well.

This table is reserved for future use and is not supported in the current release.

fcswTCPStorageStatsTable

Syntax	SEQUENCE OF FcswTCPStorageStatsEntry
Max-Access	not-accessible
Status	current
Description	A Table that lists the TCP ports and its storage statistics. This table is for future use and is not yet implemented.
Sequence	`::= { fcswPort 6 }`

fcswTCPStorageStatsEntry

Syntax	FcswTCPStorageStatsEntry
Max-Access	not-accessible
Status	current
Description	Storage statistics entries for the tcp ports.
	INDEX {fcswTCPPortIndex}

Sequence `::= { fcswTCPStorageStatsTable 1 }`

FcswTCPStorageStatsEntry ::= SEQUENCE {

fcswTCPPortReadData	Counter32,
fcswTCPPortWriteData	Counter32,
fcswTCPPortIOReads	Counter32,
fcswTCPPortIOWrites	Counter32,
fcswTCPPortPendingIOReads	Gauge32,
fcswTCPPortPendingIOWrites	Gauge32,
fcswTCPPortAvgReadCmdReqLen	Gauge32,
fcswTCPPortAvgWriteCmdReqLen	Gauge32,
fcswTCPPortUtil	INTEGER }

fcswTCPPortReadData

Syntax Counter32

Max-Access read-only

Status current

Description The total amount of actual READ throughput data (in kbytes) passed through this TCP Port since it was last initialized.

Sequence `::= { fcswTCPStorageStatsEntry 1 }`

fcswTCPPortWriteData

Syntax Counter32

Max-Access read-only

Status current

Description The total amount of actual WRITE throughput data (in kbytes) passed through this TCP Port since it was last initialized.

Sequence `::= { fcswTCPStorageStatsEntry 2 }`

fcswTCPPortIOReads

Syntax Counter32

Max-Access read-only

Status	current
Description	The total amount of IO READ operations (commands) passed through this TCP Port.
Sequence	`::= { fcswTCPStorageStatsEntry 3 }`

fcswTCPPortIOWrites

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total amount of IO Write operations (commands) passed through this TCP Port.
Sequence	`::= { fcswTCPStorageStatsEntry 4 }`

fcswTCPPortPendingIOReads

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The number of outstanding IO READ operations on this TCP port. i.e. the number of READ operations that haven't completed yet.
Sequence	`::= { fcswTCPStorageStatsEntry 5 }`

fcswTCPPortPendingIOWrites

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The number of outstanding IO WRITE operations on this TCP port. i.e. the number of WRITE operations that haven't completed yet.
Sequence	`::= { fcswTCPStorageStatsEntry 6 }`

fcswTCPPortAvgReadCmdReqLen

Syntax	Gauge32

Max-Access read-only

Status current

Description The READ command request length (in bytes) for this TCP port averaged over the last 5 secs.

Sequence `::= { fcswTCPStorageStatsEntry 7 }`

fcswTCPPortAvgWriteCmdReqLen

Syntax Gauge32

Max-Access read-only

Status current

Description The WRITE command request length (in bytes) for this TCP port averaged over the last 5 secs.

Sequence `::= { fcswTCPStorageStatsEntry 8 }`

fcswTCPPortUtil

Syntax INTEGER (1..100)

Max-Access read-only

Status current

Description A percentage value from 1-100 which provides the user a high level metric that shows how busy the switch is because of the traffic on this TCP port. This could include Q lengths, buffers available, etc.

Sequence `::= { fcswTCPStorageStatsEntry 9 }`

TCP Port Sessions Table

This table lists all the tcp sessions (initiator/target connections) for switch indexed by the TCP port and the port names of the devices. This table is reserved for future use and is not supported in the current release.

fcswNumOfTCPSessions

Syntax INTEGER (0..2147483647)

Max-Access read-only

Status current

Description The number of ongoing TCP sessions on this switch.

Sequence `::= { fcswPort 7 }`

fcswTCPSessionsTable

Syntax SEQUENCE OF FcswTCPSessionsEntry

Max-Access not-accessible

Status current

Description A Table that lists the sessions (initiator-target connections) for the TCP ports. To start traffic-statistics collection for an active session, the user could create a corresponding row with the same indices in the fcswTCPSessionStatsTable table.

When a session gets disconnected, the statistics collection is stopped and the row removed from fcswTCPSessionsTable and fcswTCPSessionStatsTable table.

This table is reserved for future use. It is not yet implemented in current releases.

Sequence `::= { fcswPort 8 }`

fcswTCPSessionsEntry

Syntax FcswTCPSessionsEntry

Max-Access not-accessible

Status current

Description Session entries for the tcp ports.

INDEX {fcswTCPPortIndex, fcswTCPSessionInitWwn, fcswTCPSessionTargetWwn }

Sequence `::= { fcswTCPSessionsTable 1 }`

FcswTCPSessionsEntry ::= SEQUENCE {

fcswTCPSessionInitWwn	WWNtype,
fcswTCPSessionTargetWwn	WWNtype,
fcswTCPSessionRemoteGwAddressType	InetAddressType,
fcswTCPSessionRemoteGwIpAddress	InetAddress }

fcswTCPSessionInitWwn

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The port WWN of the initiator (the N_Port) as registered in the fcswNsPortTable for this tcp session.
Sequence	`::= { fcswTCPSessionsEntry 1 }`

fcswTCPSessionTargetWwn

Syntax	WWNtype
Max-Access	read-only
Status	current
Description	The port WWN of the target (the N_Port) as registered in the fcswNsPortTable for this tcp session.
Sequence	`::= { fcswTCPSessionsEntry 2 }`

fcswTCPSessionRemoteGwAddressType

Syntax	InetAddressType
Max-Access	read-only
Status	current
Description	The address type for the remote gateway to which this session is connected.
Sequence	`::= { fcswTCPSessionsEntry 3 }`

fcswTCPSessionRemoteGwIpAddress

Syntax	InetAddress
Max-Access	read-only
Status	current
Description	The IP address of the remote gateway to which this session is connected.

Sequence `::= { fcswTCPSessionsEntry 4 }`

TCP Sessions Storage Statistics Table

This table lists the sessions for which storage statistics are being -- collected for.

This table is reserved for future use and is not supported in the current release.

fcswTCPSessionStatsTable

Syntax SEQUENCE OF FcswTCPSessionStatsEntry

Max-Access not-accessible

Status current

Description A Table that lists sessions and its storage statistics. To start statistics collection for a session, create a row in this table with the corresponding indices from fcswTCPSessionsTable table.

For example, if you want to start traffic collection for a session between initiator i1-WWN and target t1-WWN on port 7, then set fcswTCPSessionRowStatus to active with indices 7.i1-wwn.t1-wwn. To stop the statistics collection, then set fcswTCPSessionRowStatus to destroy.

To get the remote peer statistics, please look at rmtPeerTable in the NISHAN-GTWY.mib.

This table is for future use. It is not yet implemented in current releases.

Sequence `::= { fcswPort 9 }`

fcswTCPSessionStatsEntry

Syntax FcswTCPSessionStatsEntry

Max-Access not-accessible

Status current

Description Session storage statistics entries.

INDEX {fcswTCPPortIndex, fcswTCPSessionInitWwn, fcswTCPSessionTargetWwn }

Sequence `::= { fcswTCPSessionStatsTable 1 }`

FcswTCPSessionStatsEntry ::= SEQUENCE {

fcswTCPSessionReadData	Counter32,
fcswTCPSessionWriteData	Counter32,
fcswTCPSessionIOReads	Counter32,
fcswTCPSessionIOWrites	Counter32,
fcswTCPSessionPendingIOReads	Gauge32,
fcswTCPSessionPendingIOWrites	Gauge32,
fcswTCPSessionAvgReadCmdReqLen	Gauge32,
fcswTCPSessionAvgWriteCmdReqLen	Gauge32,
fcswTCPSessionRowStatus	RowStatus }

fcswTCPSessionReadData

Syntax Counter32

Max-Access read-only

Status current

Description The total amount of actual READ throughput data (in kbytes) passed through since this session was first initialized.

Sequence `::= { fcswTCPSessionStatsEntry 1 }`

fcswTCPSessionWriteData

Syntax Counter32

Max-Access read-only

Status current

Description The total amount of actual WRITE throughput data (in kbytes) passed through since this session was first initialized.

Sequence `::= { fcswTCPSessionStatsEntry 2 }`

fcswTCPSessionIOReads

Syntax Counter32

Max-Access read-only

Status	current
Description	The total number of IO READ operations (commands) for this TCP session.
Sequence	`::= { fcswTCPSessionStatsEntry 3 }`

fcswTCPSessionIOWrites

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The total number of IO WRITE operations (commands) for this TCP session.
Sequence	`::= { fcswTCPSessionStatsEntry 4 }`

fcswTCPSessionPendingIOReads

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The number of outstanding IO READ operations for this TCP session. i.e. the number of READ operations that haven't been completed or processed.
Sequence	`::= { fcswTCPSessionStatsEntry 5 }`

fcswTCPSessionPendingIOWrites

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The number of outstanding IO WRITE operations for this TCP session i.e. the number of WRITE operations that haven't been completed or processed.
Sequence	`::= { fcswTCPSessionStatsEntry 6 }`

fcswTCPSessionAvgReadCmdReqLen

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The READ command request length (in bytes) for this TCP session averaged over the last 5 secs.
Sequence	`::= { fcswTCPSessionStatsEntry 7 }`

fcswTCPSessionAvgWriteCmdReqLen

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The WRITE command request length (in bytes) for this TCP session averaged over the last 5 secs.
Sequence	`::= { fcswTCPSessionStatsEntry 8 }`

fcswTCPSessionRowStatus

Syntax	RowStatus
Max-Access	read-create
Status	current
Description	This object indicates the status of this entry. active (1), read-write notInService (2), read-write notReady (3), read-only createAndGo (4), write-only createAndWait (5), write-only destroy (6), write-only
Sequence	`::= { fcswTCPSessionStatsEntry 9 }`

Link Aggregation

nishanLinkAggr OBJECT IDENTIFIER ::= { nishanMgmt 10 }

laAggMaxPorts

Syntax	INTEGER (0 .. 255)
Max-Access	read-only
Status	current
Description	Maximum number of ports per aggregator.
Sequence	`::= {nishanLinkAggr 1}`

laMaxAggs

Syntax	INTEGER (0 .. 255)
Max-Access	read-only
Status	current
Description	Maximum number of aggregators per system.
Sequence	`::= {nishanLinkAggr 2}`

laConfiguredAggs

Syntax	INTEGER (0 .. 255)
Max-Access	read-only
Status	current
Description	Number of current aggregators that have been configured with ports.
Sequence	`::= {nishanLinkAggr 3}`

laAggWaitTime

Syntax	INTEGER (0..65000)
Max-Access	read-write
Status	current
Description	DURABLE: { 500 }

Wait time (in milliseconds) before starting conversation on another port. This is used if LACP and marker/responder are not implemented, and on port failure.

Sequence `::= {nishanLinkAggr 4}`

---

laAggTable

Syntax SEQUENCE OF LaAggEntry

Max-Access not-accessible

Status current

Description A Table which has list of Aggregators and its members. For software versions or hardware models that do not support Link Aggregation, this table is always empty.

Sequence `::= {nishanLinkAggr 5}`

---

laAggEntry

Syntax LaAggEntry

Max-Access not-accessible

Status current

Description Configuration information for an aggregator configured into the device by (local or network) management.

INDEX {laAggId}

Sequence `::= {laAggTable 1}`

LaAggEntry ::=

SEQUENCE {

laAggId

INTEGER,

laAggMACAddr

MacAddress,

laAggPortStatus

Integer32,

laAggAdminStatus

INTEGER,

laAggOperStatus

INTEGER,

laPhysPortList

INTEGER,

laAggName

DisplayString }

laAggId

Syntax	INTEGER (0 .. 255)
Max-Access	read-only
Status	current
Description	The ID that refers to this aggregator.
Sequence	`::= {laAggEntry 1}`

laAggMACAddr

Syntax	MacAddress
Max-Access	read-only
Status	current
Description	The MAC address of this aggregator.
Sequence	`::= {laAggEntry 2}`

laAggPortStatus

NOTE: Range removed to avoid MIB compiler errors. This should be an Unsigned32 or an octet string, but the agent returns an INTEGER. Syntax INTEGER (0 .. 4294967295)

Syntax	Integer32
Max-Access	read-only
Status	current

Description A bitmap that describes the status of the ports that are bound to this aggregator. This bitmap will be mapped to an integer in the form of 2^x where x is the position of the bit being set.

For example, an aggregator has Port 1,2 3 and 4 as members and to specify that Port 2 and Port 3 are down while port 1 and port 4 are up then we would have the value of xxxx1001(binary) 2^3 + 2^0 = 9.

Sequence `::= {laAggEntry 3}`

---

laAggAdminStatus

Syntax INTEGER {
enable(1),
disable(2),
delete(3)
}

Max-Access read-create

Status current

Description The administrative status of this aggregator. In order to delete an aggregator completely one should set delete option.

Sequence `::= {laAggEntry 4}`

---

laAggOperStatus

Syntax INTEGER { up(1), down(2) }

Max-Access read-only

Status current

Description The operation status of this aggregator.

Sequence `::= {laAggEntry 5}`

---

laPhysPortList

Syntax INTEGER (0..2147483647)

Max-Access read-create

Status current

Description A bitmap that specifies the list of ports that are bound to this aggregator. See the portStatus above in the table to see how this bitmap is mapped to an integer value.

Sequence `::= {laAggEntry 6}`

---

laAggName

Syntax DisplayString

Max-Access read-create

Status current

Description The symbolic name for this aggregator.

Sequence `::= { laAggEntry 7 }`

---

laAggAutoMode

Syntax INTEGER { enable(1), disable(2) }

Max-Access read-create

Status current

Description DURABLE: { enable }

enable/disable Link Aggregation auto detect mode.

Sequence `::= {nishanLinkAggr 6}`

END

F

McDATA SAN Routing Management MIB

FCMGMT-MIB Definitions

Version 3.0 of FA MIB

Text modified by Nishan Systems for compilation and trap integration purposes only.

10/10/02: Assigned severity levels for traps and categorized them into NNM compliant event categories.

IMPORTS

IpAddress, TimeTicks, experimental

FROM RFC1155-SMI

OBJECT-TYPE

FROM RFC-1212

DisplayString

FROM RFC1213-MIB

TRAP-TYPE

FROM RFC-1215;

Textual conventions for this MIB

FcNameId ::= OCTET STRING (SIZE(8))

FcGlobalId ::= OCTET STRING (SIZE(16))

FcAddressId ::= OCTET STRING (SIZE(3))

```
FcEventSeverity ::= INTEGER {
        unknown (1),
        emergency (2),
        alert (3),
        critical (4),
        error (5),
        warning (6),
        notify (7),
        info (8),
        debug (9),
        mark (10)  -- All messages logged
        }
FcUnitType ::= INTEGER {
unknown(1),
other(2),                   none of the following
hub(3),                     passive connectivity unit supporting loop
                            protocol.
switch(4),                  active connectivity unit supporting
                            multiple protocols.
gateway(5),                 unit that converts not only the interface
                            but also encapsulates the frame into
                            another protocol. The assumption is that
                            there is always two gateways connected
                            together. For example, FC <-> ATM.
converter(6),               unit that converts from one interface to
                            another. For example, FC <-> SCSI.
hba(7),                     host bus adapter
proxy-agent(8),             software proxy-agent
storage-device(9),          disk, cd, tape, etc.
host(10),                   host computer
storage-subsystem(11),      raid, library, etc.
module(12),                 subcomponent of a system
```

swdriver(13),	software driver
storage-access-device(14),	Provides storage management and access for heterogeneous hosts and heterogeneous devices.
wdm(15),	waveform division multiplexer
ups(16)	uninterruptable power supply }

fcmgmt OBJECT IDENTIFIER ::= { experimental 94 }

Groups in fcmgmt

connSet	OBJECT IDENTIFIER	::= { fcmgmt 1 }
trapReg	OBJECT IDENTIFIER	::= { fcmgmt 2 }
statSet	OBJECT IDENTIFIER	::= { fcmgmt 4 }
connUnitServiceSet	OBJECT IDENTIFIER	::= { fcmgmt 5 }
connUnitServiceScalars	OBJECT IDENTIFIER	::={ connUnitServiceSet 1 }
connUnitServiceTables	OBJECT IDENTIFIER	::={ connUnitServiceSet 2 }

revisionNumber

Syntax	DisplayString (SIZE (4))
Access	read-only
Status	mandatory
Description	This is the revision number for this MIB. The format of the revision value is as follows (0) = high order major revision number (1) = low order major revision number (2) = high order minor revision number (3) = low order minor revision number

The value will be stored as an ASCII value. The following is the current value of this object.

(0) = '0'

(1) = '3'

(2) = '0'

(3) = '0'

This defines a revision of 03.00

Sequence `::= { fcmgmt 3 }`

Connectivity unit group

Implementation of the group is mandatory for all systems.

uNumber

Syntax: INTEGER

Access: read-only

Status: mandatory

Description: The number of connectivity units present on this system (represented by this agent). May be a count of the boards in a chassis or the number of full boxes in a rack.

DEFVAL { 1 }

Sequence `::= { connSet 1 }`

systemURL

Syntax: DisplayString

Access: read-write

Status: mandatory

Description: The top-level URL of the system. If it does not exist the value is empty string. The URL format is implementation dependant and can have keywords embedded that are preceeded by a percent sign (e.g., %USER).

The following are the defined keywords that will be recognized and replaced with data during a launch.

USER	replace with username
PASSWORD	replace with password
GLOBALID	replace with globalid
SERIALNO	replace with serial number

If write is not supported, then return invalid. This value will be retained across boots.

DEFVAL { "" }

Sequence `::= { connSet 2 }`

---

statusChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the last status change occurred for any members of the set.

Sequence `::= { connSet 3 }`

---

configurationChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the last configuration change occurred for any members of the set. This represents a union of change information for connUnitConfigurationChangeTime.

Sequence `::= { connSet 4 }`

---

connUnitTableChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the connUnitTable was updated (an entry was either added or deleted.

Sequence `::= { connSet 5 }`

Connectivity Table

The Connectivity table contains general information on the system's units.

---

connUnitTable

Syntax SEQUENCE OF ConnUnitEntry

Access not-accessible

Status mandatory

Description A list of units under a single SNMP agent. The number of entries is given by the value of uNumber. It is 1 for stand-alone system.

Sequence `::= { connSet 6 }`

---

connUnitEntry

Syntax ConnUnitEntry

Access not-accessible

Status mandatory

Description A connectivity unit entry containing objects for a particular unit.

INDEX { connUnitId }

Sequence `::= { connUnitTable 1 }`

ConnUnitEntry ::=

SEQUENCE {

connUnitId

FcGlobalId,

connUnitGlobalId

FcGlobalId,

connUnitType

```
        FcUnitType,
connUnitNumports
        INTEGER,
connUnitState
        INTEGER,
connUnitStatus
        INTEGER,
connUnitProduct
        DisplayString,
connUnitSn
        DisplayString,
connUnitUpTime
        TimeTicks,
connUnitUrl
        DisplayString,
connUnitDomainId
        OCTET STRING,
connUnitProxyMaster
        INTEGER,
connUnitPrincipal
        INTEGER,
connUnitNumSensors
        INTEGER,
connUnitStatusChangeTime
        TimeTicks,
connUnitConfigurationChangeTime
        TimeTicks,
connUnitNumRevs
        INTEGER,
```

connUnitNumZones
INTEGER,
connUnitModuleId
FcGlobalId,
connUnitName
DisplayString,
connUnitInfo
DisplayString,
connUnitControl
INTEGER,
connUnitContact
DisplayString,
connUnitLocation
DisplayString,
connUnitEventFilter
FcEventSeverity,
connUnitNumEvents
INTEGER,
connUnitMaxEvents
INTEGER,
connUnitEventCurrID
INTEGER }

connUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The unique identification for this connectivity unit among those within this proxy domain. The value MUST be unique within the proxy domain because it is the index variable for connUnitTable.

The value assigned to a given conectivity unit SHOULD be persistent across agent and unit resets. It SHOULD be the same as connUnitGlobalId if connUnitGlobalId is known and stable.

Sequence `::= { connUnitEntry 1 }`

connUnitGlobalId

Syntax FcGlobalId

Access read-only

Status mandatory

Description An optional global-scope identifier for this connectivity unit. It MUST be a WWN for this connectivity unit or 16 octets of value zero. WWN formats requiring fewer than 16 octets MUST be extended to 16 octets with trailing zero octets, Left justified, zero filled, If a WWN is used for connUnitId, the same WWN MUST be used forconnUnitGlobalId.

When a non-zero value is provided, it SHOULD be persistent across agent and unit resets. It SHOULD be globally unique. It SHOULD be one of these FC-PH/PH3 formats:

IEEE (NAA=1)

IEEE Extended (NAA=2)

IEEE Registered (NAA=5).

IEEE Registered extended (NAA=6).

Use of the IEEE formats allows any IEEE-registered vendor to assure global uniqueness independently. The following are some references on IEEE WWN formats:

http://standards.ieee.org/regauth/oui/tutorials/fibreformat.html

http://standards.ieee.org/regauth/oui/tutorials/fibrecomp_id.html

If one or more WWNs are associated with the connUnit via other management methods, one of them SHOULD be used for connUnitGlobalId.

If there is not a WWN assigned specifically to the connUnit, there is some merit, though not a requirement, to using a WWN assigned to (one of) its permanently attached FC/LAN interface(s). This can not risk uniqueness, though.

As a counterexample, if your agent runs in a host and the host has an HBA, it is quite possible that agent, host, and HBA will all be distinct connUnits, so the host and agent can not use the WWN of the HBA.

Another example:

If your hub has a built-in Ethernet port, it might be reasonable for the hub to use its MAC address (prefixed with the appropriate NAA) as its connUnitId. But if the Ethernet were a replaceable PCCard, the hub should have an independent ID.

Sequence `::= { connUnitEntry 2 }`

---

connUnitType

Syntax FcUnitType

Access read-only

Status mandatory

Description The type of this connectivity unit.

Sequence `::= { connUnitEntry 3 }`

---

connUnitNumports

Syntax INTEGER

Access read-only

Status mandatory

Description Number of physical ports in the connectivity unit (internal/embedded, external).

Sequence `::= { connUnitEntry 4 }`

---

connUnitState

Syntax INTEGER {

unknown(1),

online(2), available for meaningful work

offline(3) unavailable for meaningful work, for example in self-test mode, configuration, etc.

}

Access read-only

Status mandatory

Description Overall state of the connectivity unit.

Sequence `::= { connUnitEntry 5 }`

---

connUnitStatus

Syntax INTEGER {

unknown(1),

unused(2), cannot report status

ok(3), available for meaningful work

warning(4), something needs attention

failed(5) something has failed

}

Access read-only

Status mandatory

Description Overall status of the connectivity unit. The goal of this object is to be the single poll point to check the status of the connunit. If there is any other component that has warning, then this should be set to warning, etc.

Sequence `::= { connUnitEntry 6 }`

---

connUnitProduct

Syntax DisplayString (SIZE (0..79))

Access read-only

Status mandatory

Description The connectivity unit vendor's product model name.

Sequence `::= { connUnitEntry 7 }`

---

connUnitSn

Syntax DisplayString (SIZE (0..79))

Access read-only

Status	mandatory
Description	The serial number for this connectivity unit.
Sequence	`::= { connUnitEntry 8 }`

connUnitUpTime

Syntax	TimeTicks
Access	read-only
Status	mandatory
Description	The number of centiseconds since the last unit initialization.
Sequence	::= { connUnitEntry 9 }

connUnitUrl

Syntax	DisplayString
Access	read-write
Status	mandatory
Description	URL to launch a management application, if applicable. Otherwise empty string. In a standalone unit, this would be the same as the top-level URL. This has the same definition as systemURL for keywords. If write is not supported, then return invalid. This value will be retained across boots.
Sequence	`::= { connUnitEntry 10 }`

connUnitDomainId

Syntax	OCTET STRING (SIZE(3))
Access	read-only
Status	mandatory
Description	24 bit Fibre Channel address ID of this connectivity unit, right justified with leading zero's if required. This should be set to the Fibre Channel address ID or if it is a switch it would be set to the Domain Controller address. If this value is not applicable, return all bits set to one.
Sequence	`::= { connUnitEntry 11 }`

connUnitProxyMaster

Syntax	INTEGER { unknown(1), no(2), yes(3) }
Access	read-only
Status	mandatory
Description	A value of 'yes' means this is the proxy master unit for a set of managed units. For example, this could be the only unit with a management card in it for a set of units. A standalone unit should return 'yes' for this object.
Sequence	`::= { connUnitEntry 12 }`

connUnitPrincipal

Syntax	INTEGER {unknown(1), no(2), yes(3) }
Access	read-only
Status	mandatory
Description	Whether this connectivity unit is the principal unit within the group of fabric elements. If this value is not applicable, return unknown.
Sequence	`::= { connUnitEntry 13 }`

connUnitNumSensors

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of sensors in the connUnitSensorTable.
Sequence	`::= { connUnitEntry 14 }`

connUnitStatusChangeTime

Syntax	TimeTicks
Access	read-only
Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the last status change occurred.

Sequence ::= { connUnitEntry 15 }

---

connUnitConfigurationChangeTime

Syntax TimeTicks

Access read-only

Status obsolete

Description The sysuptime timestamp in centiseconds at which the last configuration change occurred.

Sequence ::= { connUnitEntry 16 }

---

connUnitNumRevs

Syntax INTEGER

Access read-only

Status mandatory

Description The number of revisions in the connUnitRevsTable.

DEFVAL { 1 }

Sequence ::= { connUnitEntry 17 }

---

connUnitNumZones

Syntax INTEGER

Access read-only

Status obsolete

Description Number of zones defined in connUnitZoneTable.

Sequence ::= { connUnitEntry 18 }

---

connUnitModuleId

Syntax FcGlobalId

Access read-only

Status mandatory

Description This is a unique id, persistent between boots, that can be used to group a set of connUnits together into a module. The intended use would be to create a connUnit with a connUnitType of 'module' to represent a physical or logical group of connectivity units. Then the value of the group would be set to the value of connUnitId for this 'container' connUnit. connUnitModuleId should be zeros if this connUnit is not part of a module.

Sequence `::= { connUnitEntry 19 }`

---

connUnitName

Syntax DisplayString (SIZE(0..79))

Access read-write

Status mandatory

Description A display string containing a name for this connectivity unit. This object value should be persistent between boots.

Sequence `::= { connUnitEntry 20 }`

---

connUnitInfo

Syntax DisplayString

Access read-write

Status mandatory

Description A display string containing information about this connectivity unit. This object value should be persistent between boots.

Sequence `::= { connUnitEntry 21 }`

---

connUnitControl

Syntax INTEGER {

unknown(1),

invalid(2),

resetConnUnitColdStart(3),

resetConnUnitWarmStart(4),

offlineConnUnit(5),

onlineConnUnit(6) }

Access read-write

Status mandatory

Description This object is used to control the addressed connUnit.

NOTE: 'Cold Start' and 'Warm Start' are as defined in MIB II and are not meant to be a factory reset.

resetConnUnitColdStart	the addressed unit performs a 'Cold Start' reset.
resetConnUnitWarmStart	the addressed unit performs a 'Warm Start' reset.
offlineConnUnit	the addressed unit puts itself into an implementation dependant 'offline' state. In general, if a unit is in an offline state, it cannot be used to perform meaningful Fibre Channel work.
onlineConnUnit	the addressed unit puts itself into an implementation dependant 'online' state. In general, if a unit is in an online state, it is capable of performing meaningful Fibre Channel work.

NOTE: Each implementation may chose not to allow any or all of these values on a SET.

Sequence `::= { connUnitEntry 22 }`

connUnitContact

Syntax DisplayString (SIZE (0..79))

Access read-write

Status mandatory

Description Contact information for this connectivity unit. Persistent across boots.

Sequence `::= { connUnitEntry 23 }`

connUnitLocation

Syntax	DisplayString (SIZE (0..79))
Access	read-write
Status	mandatory
Description	Location information for this connectivity unit.Persistent across boots.
Sequence	`::= { connUnitEntry 24 }`

connUnitEventFilter

Syntax	FcEventSeverity
Access	read-write
Status	mandatory
Description	This value defines the event severity that will be logged by this connectivity unit. All events of severity less than or equal to connUnitEventFilter are logged in connUnitEventTable. Persistent across boots.
Sequence	`::= { connUnitEntry 25 }`

connUnitNumEvents

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of events currently in the connUnitEventTable.
Sequence	`::= { connUnitEntry 26 }`

connUnitMaxEvents

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Max number of events that can be defined in connUnitEventTable.

Sequence ::= { connUnitEntry 27 }

---

connUnitEventCurrID

Syntax INTEGER

Access read-only

Status mandatory

Description The last used event id (connUnitEventIndex).

Sequence ::= { connUnitEntry 28 }

The Table of revisions for hardware and software elements.

---

connUnitRevsTable

Syntax SEQUENCE OF ConnUnitRevsEntry

Access not-accessible

Status mandatory

Description Table of the revisions supported by connectivity units managed by this agent.

Sequence ::= { connSet 7 }

---

connUnitRevsEntry

Syntax ConnUnitRevsEntry

Access not-accessible

Status mandatory

Description ""

INDEX { connUnitRevsUnitId, connUnitRevsIndex }

Sequence ::= { connUnitRevsTable 1 }

ConnUnitRevsEntry ::=

SEQUENCE {

connUnitRevsUnitId

FcGlobalId,

```
connUnitRevsIndex
        INTEGER,
connUnitRevsRevId
        DisplayString,
connUnitRevsDescription
        DisplayString }
```

connUnitRevsUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this revision table.
Sequence	`::= { connUnitRevsEntry 1 }`

connUnitRevsIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitRevsEntrys with the same value of connUnitRevsUnitId, in the range between 1 and connUnitNumRevs[connUnitRevsUnitId].
Sequence	`::= { connUnitRevsEntry 2 }`

connUnitRevsRevId

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	A vendor-specific string identifying a revision of a component of the connUnit indexed by connUnitRevsUnitId.
Sequence	`::= { connUnitRevsEntry 3 }`

connUnitRevsDescription

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	Description of a component to which the revision corresponds.
Sequence	`::= { connUnitRevsEntry 4 }`

Sensor table

connUnitSensorTable

Syntax	SEQUENCE OF ConnUnitSensorEntry
Access	not-accessible
Status	mandatory
Description	Table of the sensors supported by each connectivity unit managed by this agent.
Sequence	`::= { connSet 8 }`

connUnitSensorEntry

Syntax	ConnUnitSensorEntry
Access	not-accessible
Status	mandatory
Description	Each entry contains the information for a specific sensor. INDEX { connUnitSensorUnitId, connUnitSensorIndex }

Sequence	`::= { connUnitSensorTable 1 }`
	ConnUnitSensorEntry ::=
	SEQUENCE {
	connUnitSensorUnitId
	FcGlobalId,
	connUnitSensorIndex

INTEGER (1..2147483647),
connUnitSensorName
DisplayString,
connUnitSensorStatus
INTEGER,
connUnitSensorInfo
DisplayString,
connUnitSensorMessage
DisplayString,
connUnitSensorType
INTEGER,
connUnitSensorCharacteristic
INTEGER }

connUnitSensorUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this sensor table.
Sequence	`::= { connUnitSensorEntry 1 }`

connUnitSensorIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitSensorEntrys with the same value of connUnitSensorUnitId, in the range between 1 and connUnitNumSensor[connUnitSensorUnitId].
Sequence	`::= { connUnitSensorEntry 2}`

connUnitSensorName

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	A textual identification of the sensor intended primarily for operator use.
Sequence	`::= { connUnitSensorEntry 3 }`

connUnitSensorStatus

Syntax	INTEGER {
	unknown(1),
	other(2), the sensor indicates other than ok, warning or failure.
	ok(3), the sensor indicates ok
	warning(4), the sensor indicates a warning
	failed(5) the sensor indicates failure
	}
Access	read-only
Status	mandatory
Description	The status indicated by the sensor.
Sequence	`::= { connUnitSensorEntry 4 }`

connUnitSensorInfo

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	Miscellaneous static info about the sensor such as its serial number.
Sequence	`::= { connUnitSensorEntry 5 }`

connUnitSensorMessage

Syntax	DisplayString

Access read-only

Status mandatory

Description This describes the status of the sensor as a message. It may also provide more resolution on the sensor indication, for example 'Cover temperature 1503K, above nominal operating range'

Sequence `::= { connUnitSensorEntry 6 }`

---

connUnitSensorType

Syntax INTEGER {

unknown(1),
other(2),
battery(3),
fan(4),
power-supply(5),
transmitter(6),
enclosure(7),
board(8),
receiver(9) }

Access read-only

Status mandatory

Description The type of component being monitored by this sensor.

Sequence `::= { connUnitSensorEntry 7 }`

---

connUnitSensorCharacteristic

Syntax INTEGER {

unknown(1),
other(2),
temperature(3),
pressure(4),
emf(5),

currentValue(6), -- current is a keyword
airflow(7),
frequency(8),
power(9),
door(10) }

Access read-only

Status mandatory

Description The characteristics being monitored by this sensor.

Sequence `::= { connUnitSensorEntry 8 }`

Port Table

connUnitPortTable

Syntax SEQUENCE OF ConnUnitPortEntry

Access not-accessible

Status mandatory

Description Generic information on ports for a specific connUnit.

Sequence `::= { connSet 10 }`

connUnitPortEntry

Syntax ConnUnitPortEntry

Access not-accessible

Status mandatory

Description Each entry contains the information for a specific port.
INDEX { connUnitPortUnitId, connUnitPortIndex }

Sequence `::= { connUnitPortTable 1 }`
ConnUnitPortEntry ::=

SEQUENCE {	
connUnitPortUnitId	FcGlobalId,
connUnitPortIndex	INTEGER,
connUnitPortType	INTEGER,
connUnitPortFCClassCap	OCTET STRING,
connUnitPortFCClassOp	OCTET STRING,
connUnitPortState	INTEGER,
connUnitPortStatus	INTEGER,
connUnitPortTransmitterType	INTEGER,
connUnitPortModuleType	INTEGER,
connUnitPortWwn	FcNameId,
connUnitPortFCId	FcAddressId,
connUnitPortSn	DisplayString,
connUnitPortRevision	DisplayString,
connUnitPortVendor	DisplayString,
connUnitPortSpeed	INTEGER,
connUnitPortControl	INTEGER,
connUnitPortName	DisplayString,
connUnitPortPhysicalNumber	INTEGER,
connUnitPortStatObject	OBJECT IDENTIFIER,
connUnitPortProtocolCap	OCTET STRING,
connUnitPortProtocolOp	OCTET STRING,
connUnitPortNodeWwn	FcNameId,
connUnitPortHWState	INTEGER }

connUnitPortUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this port.
Sequence	`::= { connUnitPortEntry 1 }`

connUnitPortIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitPortEntrys on this connectivity unit, between 1 and connUnitNumPort[connUnitPortUnitId].
Sequence	`::= { connUnitPortEntry 2 }`

connUnitPortType

Syntax	INTEGER {	
	unknown (1),	
	other (2),	
	not-present (3),	
	hub-port (4),	
	n-port (5),	end port for fabric
	nl-port (6),	end port for loop
	fl-port (7),	public loop
	f-port (8),	fabric port
	e-port (9),	fabric expansion port
	g-port (10),	generic fabric port
	domain-ctl (11),	domain controller
	hub-controller(12),	
	scsi (13),	parallel SCSI port
	escon (14),	
	lan (15),	
	wan (16),	
	ac (17),	AC power line
	dc (18),	DC power line
	ssa (19)	serial storage architecture }
Access	read-only	
Status	mandatory	

Description The port type.

Sequence `::= { connUnitPortEntry 3 }`

connUnitPortFCClassCap

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the classes of service capability of this port. If this is not applicable, return all bits set to zero.

The bits have the following definition:

unknown	0
class-f	1
class-one	2
class-two	4
class-three	8
class-four	16
class-five	32
class-six	64

Sequence `::= { connUnitPortEntry 4 }`

connUnitPortFCClassOp

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the classes of service that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortFCClassCap.

Sequence `::= { connUnitPortEntry 5 }`

connUnitPortState

Syntax	INTEGER {	
	unknown(1),	
	online(2),	available for meaningful work
	offline(3),	not available for meaningful work
	bypassed(4),	no longer used (4/12/00)
	diagnostics(5)	
	}	
Access	read-only	
Status	mandatory	
Description	The user selected state of the port hardware.	
Sequence	`::= { connUnitPortEntry 6 }`	

connUnitPortStatus

Syntax	INTEGER {	
	unknown (1),	
	unused (2),	device cannot report this status
	ready (3),	FCAL Loop or FCPH Link reset protocol initialization has completed
	warning (4),	do not use (4/12/00)
	failure (5),	do not use (4/12/00)
	notparticipating (6),	loop notparticipating and does not have a loop address
	initializing (7),	protocol is proceeding
	bypass (8),	do not use (4/12/00)
	ols (9)	FCP offline status }
Access	read-only	
Status	mandatory	
Description	An overall protocol status for the port. This value of connUnitPortState is not online, then this is reported Unknown.	
Sequence	`::= { connUnitPortEntry 7 }`	

connUnitPortTransmitterType

Syntax INTEGER {
unknown(1),
other(2),
unused(3),
shortwave(4),
longwave(5),
copper(6),
scsi(7),
longwaveNoOFC(8),
shortwaveNoOFC(9),
longwaveLED(10),
ssa(11) }

Access read-only

Status mandatory

Description The technology of the port transceiver.

Sequence `::= { connUnitPortEntry 8 }`

connUnitPortModuleType

Syntax INTEGER {
unknown(1),
other(2),
gbic(3),
embedded(4), -- fixed, i.e., oneXnine
glm(5),
gbicSerialId(6),
gbicNoSerialId(7),
gbicNotInstalled(8),

smallFormFactor(9) -- this is generically a small form factor connector.

}

Access read-only

Status mandatory

Description The module type of the port connector.

Sequence `::= { connUnitPortEntry 9 }`

---

connUnitPortWwn

Syntax FcNameId

Access read-only

Status mandatory

Description The World Wide Name of the port if applicable, otherwise all zeros.

Sequence `::= { connUnitPortEntry 10 }`

---

connUnitPortFCId

Syntax FcAddressId

Access read-only

Status mandatory

Description This is the assigned Fibre Channel ID of this port. This value is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected.

If this is an eport, then it will only contain the domain ID left justified, zero filled. If this port does not have a Fibre Channel address, return all bits set to 1.

Sequence `::= { connUnitPortEntry 11 }`

---

connUnitPortSn

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The serial number of the unit (e.g., for a GBIC). If this is not applicable, return empty string.

Sequence `::= { connUnitPortEntry 12 }`

---

connUnitPortRevision

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The port revision (e.g., for a GBIC).

Sequence `::= { connUnitPortEntry 13 }`

---

connUnitPortVendor

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The port vendor (e.g., for a GBIC).

Sequence `::= { connUnitPortEntry 14 }`

---

connUnitPortSpeed

Syntax INTEGER

Access read-only

Status mandatory

Description The speed of the port in kilobytes per second.

Sequence `::= { connUnitPortEntry 15 }`

---

connUnitPortControl

Syntax INTEGER {

unknown(1),

invalid(2),

resetConnUnitPort(3),

bypassConnUnitPort(4),
unbypassConnUnitPort(5),
offlineConnUnitPort(6),
onlineConnUnitPort(7),
resetConnUnitPortCounters(8)
}

Access read-write

Status mandatory

Description

This object is used to control the addressed connUnit's port. Valid commands are:

resetConnUnitPort:	If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'reset' operation. Examples of these operations are: the Link Reset protocol, the Loop Initialization protocol, or a resynchronization occurring between the transceiver in the addressed port to the transceiver that the port is connected to.
bypassConnUnitPort:	If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'bypass' operation. Examples of these operations are transitioning from online to offline, a request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.
unbypassConnUnitPort:	If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'unbypass' operation. Examples of these operations are the Link Failure protocol, a request(PARTICIPATING) command to the Loop Port state machine, or addition of the port to an arbitrated loop by a hub.

offlineConnUnitPort: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'offline' operation. Examples of these operations are disabling a port's transceiver, the Link Failure protocol, request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.

onlineConnUnitPort: If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'online' operation. Examples of these operations are enabling a port's transceiver, the Link Failure protocol, request (PARTICIPATING) command to the Loop Port state machine, or addition of the port from an arbitrated loop by a hub.

resetConnUnitPortCounters: If the addressed connUnit allows this operation to be performed to this port, the addressed port statistics table counters will be set to zero.

NOTE: Each implementation may chose not to allow any or all of these values on a SET. On a read, if you do not support write, then return invalid. Otherwise return the last control operation attempted.

Sequence `::= { connUnitPortEntry 16 }`

connUnitPortName

Syntax DisplayString

Access read-write

Status mandatory

Description A user-defined name for this port. This means that up to DisplayString characters may be supported. If less than, then the name will be truncated in the connunit.

Sequence `::= { connUnitPortEntry 17 }`

connUnitPortPhysicalNumber

Syntax INTEGER

Access read-only

Status mandatory

Description This is the internal port number this port is known by. In many implementations, this should be the same as connUnitPortIndex. Some implementations may have an internal port representation not compatible with the rules for table indices. In that case, provide the internal representation of this port in this object. This value may also be used in the connUnitLinkPortNumberX or connUnitLinkPortNumberY objects of the connUnitLinkTable.

Sequence `::= { connUnitPortEntry 18 }`

connUnitPortStatObject

Syntax OBJECT IDENTIFIER

Access read-only

Status deprecated

Description This contains the OID of the first object of the table that contains the statistics for this particular port. If this has a value of zero, then there are no statistics available for this port. The port type information will help identify the statistics objects that will be found in the table.

Sequence `::= { connUnitPortEntry 19 }`

connUnitPortProtocolCap

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the driver level protocol capability of this port. If this is not applicable, return all bits set to zero.

The bits have the following definition:

unknown - 0

Loop - 1

Fabric - 2

SCSI - 4

TCP/IP - 8

VI - 16

FICON - 32

Sequence `::= { connUnitPortEntry 20 }`

---

connUnitPortProtocolOp

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the driver level protocol(s) that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortProtocolCap.

Sequence `::= { connUnitPortEntry 21 }`

---

connUnitPortNodeWwn

Syntax FcNameId

Access read-only

Status mandatory

Description The Node World Wide Name of the port if applicable, otherwise all zeros. This should have the same value for a group of related ports. The container is defined as the largest physical entity. For example, all ports on HBAs on a host will have the same Node WWN. All ports on the same storage subsystem will have the same Node WWN.

Sequence `::= { connUnitPortEntry 22 }`

connUnitPortHWState

Syntax	INTEGER {	
	unknown (1),	
	failed (2),	port failed diagnostics
	bypassed (3),	FCAL bypass, loop only
	active (4),	connected to a device
	loopback (5),	Port in ext loopback
	txfault (6),	Transmitter fault
	noMedia (7),	media not installed
	linkDown (8)	waiting for activity (rx sync) }
Access	read-only	
Status	mandatory	
Description	The hardware detected state of the port.	
Sequence	`::= { connUnitPortEntry 23 }`	

Event Group

connUnitEventTable

Syntax	SEQUENCE OF ConnUnitEventEntry
Access	not-accessible
Status	mandatory
Description	The table of connectivity unit events. Errors, warnings, and information should be reported in this table.
Sequence	`::= { connSet 11 }`

connUnitEventEntry

Syntax	ConnUnitEventEntry
Access	not-accessible
Status	mandatory

Description — Each entry contains information on a specific event for the given connectivity unit.

INDEX { connUnitEventUnitId, connUnitEventIndex }

Sequence

```
::= { connUnitEventTable 1 }
ConnUnitEventEntry ::=
SEQUENCE {
connUnitEventUnitId
        FcGlobalId,
connUnitEventIndex
        INTEGER (1..2147483647),
connUnitEventId
        INTEGER,
connUnitREventTime
        DisplayString,
connUnitSEventTime
        TimeTicks,
connUnitEventSeverity
        FcEventSeverity,
connUnitEventType
        INTEGER,
connUnitEventObject
        OBJECT IDENTIFIER,
connUnitEventDescr
        DisplayString }
```

connUnitEventUnitId

Syntax — FcGlobalId

Access — read-only

Status — mandatory

Description — The connUnitId of the connectivity unit that contains this event table.

Sequence `::= { connUnitEventEntry 1 }`

connUnitEventIndex

Syntax INTEGER (1..2147483647)

Access read-only

Status mandatory

Description Each connectivity unit has its own event buffer. As it wraps, it may write over previous events. This object is an index into the buffer. It is recommended that this table be read using 'getNext's to retrieve the initial table. The management application should read the event table at periodic intervals and then determine if any new entries were added by comparing the last known index value with the current highest index value. The management application should then update its copy of the event table. If the read interval is too long, it is possible that there may be events that may not be contained in the agent's internal event buffer.

For example, an agent may read events 50-75. At the next read interval, connUnitEventCurrID is 189. If the management app tries to read event index 76, and the agent's internal buffer is 100 entries max, event index 76 will no longer be available.

The index value is an incrementing integer starting from one every time there is a table reset. On table reset, all contents are emptied and all indices are set to zero. When an event is added to the table, the event is assigned the next higher integer value than the last item entered into the table. If the index value reaches its maximum value, the next item entered will cause the index value to roll over and start at one again.

Sequence `::= { connUnitEventEntry 2 }`

connUnitEventId

Syntax INTEGER

Access read-only

Status deprecated

Description The internal event Id. Incremented for each event, ranging between 1 and connUnitMaxEvents. Not used as table index to simplify the agent implementation. When this reaches the end of the range

specified by connUnitMaxEvents, the Id will roll over to start at one. This value will be set back to one at reset. The relationship of this value to the index is that internal event id may represent a smaller number than a 32 bit integer (e.g. max 100 entries) and would only have a value range up to connUnitMaxEvents.

Sequence `::= { connUnitEventEntry 3 }`

---

connUnitREventTime

Syntax DisplayString (SIZE (0..15))

Access read-only

Status mandatory

Description This is the real time when the event occurred. It has the following format.

DDMMYYYY HHMMSS

DD=day number

MM=month number

YYYY=year number

HH=hour number

MM=minute number

SS=seconds number

If not applicable, return either a NULL string or '00000000 000000'.

Sequence `::= { connUnitEventEntry 4 }`

---

connUnitSEventTime

Syntax TimeTicks

Access read-only

Status mandatory

Description This is the sysuptime timestamp when the event occurred.

Sequence `::= { connUnitEventEntry 5 }`

connUnitEventSeverity

Syntax	FcEventSeverity
Access	read-only
Status	mandatory
Description	The event severity level.
Sequence	`::= { connUnitEventEntry 6 }`

connUnitEventType

Syntax	INTEGER { unknown(1), other(2), status(3), configuration(4), topology(5) }
Access	read-only
Status	mandatory
Description	The type of this event.
Sequence	`::= { connUnitEventEntry 7 }`

connUnitEventObject

Syntax	OBJECT IDENTIFIER
Access	read-only
Status	mandatory
Description	This is used with the connUnitEventType to identify which object the event refers to. Examples are connUnitPortStatus.connUnitId.connUnitPortIndex, connUnitStatus.connUnitId, etc.
Sequence	`::= { connUnitEventEntry 8 }`

connUnitEventDescr

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	The description of the event.
Sequence	`::= { connUnitEventEntry 9 }`

Link Table

This is intended to organize and communicate any information the agent possesses which would assist a management application to discover the CONNECTIVITY UNITS in the framework and the TOPOLOGY of their interconnect. That is, the goal is to assist the management application not only to LIST the elements of the framework, but to MAP them.

With this goal, the agent SHOULD include as much as it possesses about any links from its own connectivity units to others, including links among its own units.

An agent SHOULD include partial information about links if it is not able to fully define them. For an entry to be considered to be valid, both the X (local) and the Y (remote) need to have one valid value.

If the agent is able to discover links which do not directly attach to members of its agency and its discovery algorithm gives some assurance the links are recently valid, it MAY include these links.

Link information entered by administrative action MAY be included even if not validated directly if the link has at least one endpoint in this agency, but SHOULD NOT be included otherwise.

A connectivity unit should fill the table in as best it can. One of the methods to fill this in would be to use the RNID ELS (ANSI document 99-422v0). This allows one to query a port for the information needed for the link table.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table.

connUnitLinkTable

Syntax	SEQUENCE OF ConnUnitLinkEntry
Access	not-accessible
Status	mandatory
Description	A list of links know to this agent from this connectivity unit to other connectivity units.
Sequence	`::= { connSet 12 }`

connUnitLinkEntry

Syntax	ConnUnitLinkEntry
Access	not-accessible
Status	mandatory
Description	An entry describing a particular link to another. INDEX { connUnitLinkUnitId, connUnitLinkIndex }
Sequence	`::= { connUnitLinkTable 1 }`

ConnUnitLinkEntry ::=

SEQUENCE {

connUnitLinkUnitId	FcGlobalId,
connUnitLinkIndex	INTEGER,
connUnitLinkNodeIdX	OCTET STRING,
connUnitLinkPortNumberX	INTEGER,
connUnitLinkPortWwnX	FcGlobalId,
connUnitLinkNodeIdY	OCTET STRING,
connUnitLinkPortNumberY	INTEGER,
connUnitLinkPortWwnY	FcGlobalId,
connUnitLinkAgentAddressY	OCTET STRING,
connUnitLinkAgentAddressTypeY	INTEGER,
connUnitLinkAgentPortY	INTEGER,

```
                    connUnitLinkUnitTypeY       FcUnitType,
                    connUnitLinkConnIdY         OCTET STRING,
                    connUnitLinkCurrIndex       INTEGER }
```

connUnitLinkUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this link table.
Sequence	`::= { connUnitLinkEntry 1 }`

connUnitLinkIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	This index is used to create a unique value for each entry in the link table with the same connUnitLinkUnitId. The value can only be reused if it is not currently in use and the value is the next candidate to be used. This value wraps at the highest value represented by the size of INTEGER. This value is reset to zero when the system is reset and the first value to be used is one.
Sequence	`::= { connUnitLinkEntry 2 }`

connUnitLinkNodeIdX

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The node WWN of the unit at one end of the link. If the node WWN is unknown and the node is a connUnit in the responding agent then the value of this object MUST BE equal to its connUnitID.
Sequence	`::= { connUnitLinkEntry 3 }`

connUnitLinkPortNumberX

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The port number on the unit specified by connUnitLinkNodeIdX if known, otherwise -1. If the value is nonnegative then it will be equal to connUnitPortPhysicalNumber.
Sequence	`::= { connUnitLinkEntry 4 }`

connUnitLinkPortWwnX

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The port WWN of the unit specified by connUnitLinkNodeIdX if known, otherwise 16 octets of binary 0.
Sequence	`::= { connUnitLinkEntry 5 }`

connUnitLinkNodeIdY

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The node WWN of the unit at the other end of the link. If the node WWN is unknown and the node is a connUnit in the responding SNMP agency then the value of this object MUST BE equal to its connUnitID.
Sequence	`::= { connUnitLinkEntry 6 }`

connUnitLinkPortNumberY

Syntax	INTEGER
Access	read-only
Status	mandatory

Description The port number on the unit specified by connUnitLinkNodeIdY if known, otherwise -1. If the value is nonnegative then it will be equal to connUnitPortPhysicalNumber.

Sequence `::= { connUnitLinkEntry 7 }`

---

connUnitLinkPortWwnY

Syntax FcGlobalId

Access read-only

Status mandatory

Description The port WWN on the unit specified by connUnitLinkNodeIdY if known, otherwise 16 octets of binary 0.

Sequence `::= { connUnitLinkEntry 8 }`

---

connUnitLinkAgentAddressY

Syntax OCTET STRING (SIZE(16))

Access read-only

Status mandatory

Description The address of an FCMGMT MIB agent for the node identified by connUnitLinkNodeIdY, if known; otherwise 16 octets of binary 0.

Sequence `::= { connUnitLinkEntry 9 }`

---

connUnitLinkAgentAddressTypeY

Syntax INTEGER

Access read-only

Status mandatory

Description If connUnitLinkAgentAddressY is nonzero, it is a protocol address. ConnUnitLinkAgentAddressTypeY is the 'address family number' assigned by IANA to identify the address format. (e.g., 1 is Ipv4, 2 is Ipv6). If connUnitLinkAgentAddressY is all zeros, then this value is ignored.

Sequence `::= { connUnitLinkEntry 10 }`

connUnitLinkAgentPortY

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The IP port number for the agent. This is provided in case the agent is at a non-standard SNMP port.
Sequence	`::= { connUnitLinkEntry 11 }`

connUnitLinkUnitTypeY

Syntax	FcUnitType
Access	read-only
Status	mandatory
Description	Type of the FC connectivity unit as defined in connUnitType.
Sequence	`::= { connUnitLinkEntry 12 }`

connUnitLinkConnIdY

Syntax	OCTET STRING (SIZE(3))
Access	read-only
Status	mandatory
Description	This is the Fibre Channel ID of this port. If the connectivity unit is a switch, this is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected. If this is an eport, then it will only contain the domain ID. If not any of those, unknown or cascaded loop, return all bits set to 1.
Sequence	`::= { connUnitLinkEntry 13 }`

connUnitLinkCurrIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The last used link index.

Sequence `::= { connUnitLinkEntry 14 }`

The following four tables have been obsoleted. These were used to keep statistic information based on the type of port type. It was changed for all ports to use a common statistics table.

---

connUnitPortStatHubTable

Syntax SEQUENCE OF ConnUnitPortStatHubEntry

Access not-accessible

Status obsolete

Description A list of statistics for the hub port type.

Sequence `::= { statSet 1 }`

---

connUnitPortStatFabricTable

Syntax SEQUENCE OF ConnUnitPortStatFabricEntry

Access not-accessible

Status obsolete

Description A list of statistics for the fabric port types.

Sequence `::= { statSet 2 }`

---

connUnitPortStatSCSITable

Syntax SEQUENCE OF ConnUnitPortStatSCSIEntry

Access not-accessible

Status obsolete

Description A list of statistics for the SCSI port type.

Sequence `::= { statSet 3 }`

---

connUnitPortStatLANTable

Syntax SEQUENCE OF ConnUnitPortStatLANEntry

Access not-accessible

Status obsolete

Description A list of statistics for the LAN/WAN port type.

Sequence `::= { statSet 4 }`

There is one and only one statistics table for each individual port. For all objects in statistics table, if the object is not supported by the conn unit then the high order bit is set to 1 with all other bits set to zero. The high order bit is reserved to indicate if the object if supported or not. All objects start at a value of zero at hardware initialization and continue incrementing till end of 63 bits and then wrap to zero.

Port Statistics

connUnitPortStatTable

Syntax SEQUENCE OF ConnUnitPortStatEntry

Access not-accessible

Status mandatory

Description A list of statistics for the fabric port types.

Sequence `::= { statSet 5 }`

connUnitPortStatEntry

Syntax ConnUnitPortStatEntry

Access not-accessible

Status mandatory

Description An entry describing port statistics.

INDEX { connUnitPortStatUnitId, connUnitPortStatIndex }

Sequence `::= { connUnitPortStatTable 1 }`

ConnUnitPortStatEntry ::=

SEQUENCE {

connUnitPortStatUnitId

FcGlobalId,

connUnitPortStatIndex

INTEGER,

connUnitPortStatCountError
OCTET STRING,
connUnitPortStatCountTxObjects
OCTET STRING,
connUnitPortStatCountRxObjects
OCTET STRING,
connUnitPortStatCountTxElements
OCTET STRING,
connUnitPortStatCountRxElements
OCTET STRING,
connUnitPortStatCountBBCreditZero
OCTET STRING,
connUnitPortStatCountInputBuffersFull
OCTET STRING,
connUnitPortStatCountFBSYFrames
OCTET STRING,
connUnitPortStatCountPBSYFrames
OCTET STRING,
connUnitPortStatCountFRJTFrames
OCTET STRING,
connUnitPortStatCountPRJTFrames
OCTET STRING,
connUnitPortStatCountClass1RxFrames
OCTET STRING,
connUnitPortStatCountClass1TxFrames
OCTET STRING,
connUnitPortStatCountClass1FBSYFrames
OCTET STRING,
connUnitPortStatCountClass1PBSYFrames

OCTET STRING,
connUnitPortStatCountClass1FRJTFrames
OCTET STRING,
connUnitPortStatCountClass1PRJTFrames
OCTET STRING,
connUnitPortStatCountClass2RxFrames
OCTET STRING,
connUnitPortStatCountClass2TxFrames
OCTET STRING,
connUnitPortStatCountClass2FBSYFrames
OCTET STRING,
connUnitPortStatCountClass2PBSYFrames
OCTET STRING,
connUnitPortStatCountClass2FRJTFrames
OCTET STRING,
connUnitPortStatCountClass2PRJTFrames
OCTET STRING,
connUnitPortStatCountClass3RxFrames
OCTET STRING,
connUnitPortStatCountClass3TxFrames
OCTET STRING,
connUnitPortStatCountClass3Discards
OCTET STRING,
connUnitPortStatCountRxMulticastObjects
OCTET STRING,
connUnitPortStatCountTxMulticastObjects
OCTET STRING,
connUnitPortStatCountRxBroadcastObjects
OCTET STRING,

connUnitPortStatCountTxBroadcastObjects
OCTET STRING,
connUnitPortStatCountRxLinkResets
OCTET STRING,
connUnitPortStatCountTxLinkResets
OCTET STRING,
connUnitPortStatCountNumberLinkResets
OCTET STRING,
connUnitPortStatCountRxOfflineSequences
OCTET STRING,
connUnitPortStatCountTxOfflineSequences
OCTET STRING,
connUnitPortStatCountNumberOfflineSequences
OCTET STRING,
connUnitPortStatCountLinkFailures
OCTET STRING,
connUnitPortStatCountInvalidCRC
OCTET STRING,
connUnitPortStatCountInvalidTxWords
OCTET STRING,
connUnitPortStatCountPrimitiveSequenceProtocolErrors
OCTET STRING,
connUnitPortStatCountLossofSignal
OCTET STRING,
connUnitPortStatCountLossofSynchronization
OCTET STRING,
connUnitPortStatCountInvalidOrderedSets
OCTET STRING,
connUnitPortStatCountFramesTooLong

OCTET STRING,
connUnitPortStatCountFramesTruncated
OCTET STRING,
connUnitPortStatCountAddressErrors
OCTET STRING,
connUnitPortStatCountDelimiterErrors
OCTET STRING,
connUnitPortStatCountEncodingDisparityErrors
OCTET STRING }

connUnitPortStatUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this port stat table.
Sequence	`::= { connUnitPortStatEntry 1 }`

connUnitPortStatIndex

Syntax	INTEGER (0..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all entries in this table, between 0 and connUnitNumPort[connUnitPortUnitId].
Sequence	`::= { connUnitPortStatEntry 2 }`

connUnitPortStatCountError

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description A count of the errors that have occurred on this port.

Sequence `::= { connUnitPortStatEntry 3 }`

---

connUnitPortStatCountTxObjects

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description The number of frames/packets/IOs/etc. that have been transmitted by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Tx objects.

Sequence `::= { connUnitPortStatEntry 4 }`

---

connUnitPortStatCountRxObjects

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description The number of frames/packets/IOs/etc. that have been received by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Rx objects.

Sequence `::= { connUnitPortStatEntry 5 }`

---

connUnitPortStatCountTxElements

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description The number of octets or bytes that have been transmitted by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.

Sequence `::= { connUnitPortStatEntry 6 }`

connUnitPortStatCountRxElements

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of octets or bytes that have been received. by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.
Sequence	`::= { connUnitPortStatEntry 7 }`

connUnitPortStatCountBBCreditZero

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of transitions in/out of BBcredit zero state. The other side is not providing any credit. Note, this is a Fibre Channel stat only.
Sequence	`::= { connUnitPortStatEntry 8 }`

connUnitPortStatCountInputBuffersFull

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of occurrences when all input buffers of a port were full and outbound buffer-to-buffer credit transitioned to zero. There is no credit to provide to other side. Note, this is a Fibre Channel stat only.
Sequence	`::= { connUnitPortStatEntry 9 }`

connUnitPortStatCountFBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description Count of times that FBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. Port can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

Sequence `::= { connUnitPortStatEntry 10 }`

---

connUnitPortStatCountPBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that PBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if the destination port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

Sequence `::= { connUnitPortStatEntry 11 }`

---

connUnitPortStatCountFRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Frame that was rejected by the fabric. Note, This is the total for all classes and is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 12 }`

---

connUnitPortStatCountPRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Frame that was rejected at the destination N_Port. Note, This is the total for all classes and is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 13 }`

---

connUnitPortStatCountClass1RxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 1 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 14 }`

---

connUnitPortStatCountClass1TxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 1 Frames transmitted out this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 15 }`

---

connUnitPortStatCountClass1FBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 16 }`

connUnitPortStatCountClass1PBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that PBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 17 }`

connUnitPortStatCountClass1FRJTFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 18 }`

connUnitPortStatCountClass1PRJTFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 19 }`

connUnitPortStatCountClass2RxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only

Status mandatory

Description Count of Class 2 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 20 }`

---

connUnitPortStatCountClass2TxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 2 Frames transmitted out this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 21 }`

---

connUnitPortStatCountClass2FBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 22 }`

---

connUnitPortStatCountClass2PBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that PBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 23 }

---

connUnitPortStatCountClass2FRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 24 }

---

connUnitPortStatCountClass2PRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 25 }

---

connUnitPortStatCountClass3RxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 3 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 26 }

---

connUnitPortStatCountClass3TxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 3 Frames transmitted out this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 27 }`

---

connUnitPortStatCountClass3Discards

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 3 Frames that were discarded upon reception at this port. There is no FBSY or FRJT generated for Class 3 Frames. They are simply discarded if they cannot be delivered. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 28 }`

---

connUnitPortStatCountRxMulticastObjects

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Multicast Frames or Packets received at this port.

Sequence `::= { connUnitPortStatEntry 29 }`

---

connUnitPortStatCountTxMulticastObjects

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Multicast Frames or Packets transmitted out this port.

Sequence `::= { connUnitPortStatEntry 30 }`

---

connUnitPortStatCountRxBroadcastObjects

Syntax OCTET STRING (SIZE (8))

Access	read-only
Status	mandatory
Description	Count of Broadcast Frames or Packets received at this port.
Sequence	`::= { connUnitPortStatEntry 31 }`

connUnitPortStatCountTxBroadcastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Broadcast Frames or Packets transmitted out this port. On a Fibre Channel loop, count only OPNr frames generated.
Sequence	`::= { connUnitPortStatEntry 32 }`

connUnitPortStatCountRxLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets. This is the number of LRs received. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 33 }`

connUnitPortStatCountTxLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets. This is the number LRs transmitted. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 34 }`

connUnitPortStatCountNumberLinkResets

Syntax	OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Link resets and LIPs detected at this port. The number times the reset link protocol is initiated. These are the count of the logical resets, a count of the number of primatives. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 35 }`

---

connUnitPortStatCountRxOfflineSequences

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Offline Primitive OLS received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 36 }`

---

connUnitPortStatCountTxOfflineSequences

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Offline Primitive OLS transmitted by this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 37 }`

---

connUnitPortStatCountNumberOfflineSequences

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Offline Primitive sequence received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 38 }`

connUnitPortStatCountLinkFailures

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of link failures. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 39 }`

connUnitPortStatCountInvalidCRC

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received with invalid CRC. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Loop ports should not count CRC errors passing through when monitoring. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 40 }`

connUnitPortStatCountInvalidTxWords

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of invalid transmission words received at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 41 }`

connUnitPortStatCountPrimitiveSequenceProtocolErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description Count of primitive sequence protocol errors detected at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 42 }`

---

connUnitPortStatCountLossofSignal

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of instances of signal loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 43 }`

---

connUnitPortStatCountLossofSynchronization

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of instances of synchronization loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 44 }`

---

connUnitPortStatCountInvalidOrderedSets

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of invalid ordered sets received at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 45 }`

connUnitPortStatCountFramesTooLong

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received at this port where the frame length was greater than what was agreed to in FLOGI/PLOGI. This could be caused by losing the end of frame delimiter. Note, this is a Fibre Channel only stat.
Sequence	`::= { connUnitPortStatEntry 46 }`

connUnitPortStatCountFramesTruncated

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received at this port where the frame length was less than the minimum indicated by the frame header - normally 24 bytes, but it could be more if the DFCTL field indicates an optional header should have been present. Note, this is a FC only stat.
Sequence	`::= { connUnitPortStatEntry 47 }`

connUnitPortStatCountAddressErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received with unknown addressing. e.x. unknown SID or DID. the SID or DID is not known to the routing algorithm. Note. this is a FC only stat.
Sequence	`::= { connUnitPortStatEntry 48 }`

connUnitPortStatCountDelimiterErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only

Status mandatory

Description Count of invalid frame delimiters received at this port. An example is a frame with a class 2 start and a class 3 at the end. Note, this is a FC only stat.

Sequence `::= { connUnitPortStatEntry 49 }`

connUnitPortStatCountEncodingDisparityErrors

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of disparity errors received at this port. Note, this is a Fibre Channel only stat.

Sequence `::= { connUnitPortStatEntry 50 }`

FC Simple Name Server Table

The Fibre Channel Simple Name Server table contains an entry for each device presently known to this connUnit. There will not be any version on this since FC-GS3 does not define a version today.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table

connUnitSnsMaxEntry

Syntax INTEGER

Access read-only

Status mandatory

Description The maximum number of entries in the table.

Sequence `::= { connUnitServiceScalars 1 }`

connUnitSnsTable

Syntax	SEQUENCE OF ConnUnitSnsEntry
Access	not-accessible
Status	mandatory
Description	This table contains an entry for each object registered with this port in the switch.
Sequence	`::= { connUnitServiceTables 1 }`

connUnitSnsEntry

Syntax	ConnUnitSnsEntry
Access	not-accessible
Status	mandatory
Description	The Simple Name Server table for the port represented by ConnUnitSnsPortIndex . INDEX { connUnitSnsId, connUnitSnsPortIndex }

Sequence	`::= { connUnitSnsTable 1 }`
	ConnUnitSnsEntry ::=
	SEQUENCE {
	connUnitSnsId
	OCTET STRING,
	connUnitSnsPortIndex
	INTEGER,
	connUnitSnsPortIdentifier
	FcAddressId,
	connUnitSnsPortName
	FcNameId,
	connUnitSnsNodeName
	FcNameId,
	connUnitSnsClassOfSvc

OCTET STRING,
connUnitSnsNodeIPAddress
OCTET STRING,
connUnitSnsProcAssoc
OCTET STRING,
connUnitSnsFC4Type
OCTET STRING,
connUnitSnsPortType
OCTET STRING,
connUnitSnsPortIPAddress
OCTET STRING,
connUnitSnsFabricPortName
FcNameId,
connUnitSnsHardAddress
FcAddressId,
connUnitSnsSymbolicPortName
DisplayString,
connUnitSnsSymbolicNodeName
DisplayString }

---

connUnitSnsId

Syntax	OCTET STRING (SIZE (16))
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this Name Server table.
Sequence	`::= { connUnitSnsEntry 1 }`

---

connUnitSnsPortIndex

Syntax	INTEGER

Access	read-only
Status	mandatory
Description	The physical port number of this SNS table entry. Each physical port has an SNS table with 1-n entries indexed by ConnUnitSnsPortIdentifier (port address).
Sequence	`::= { connUnitSnsEntry 2 }`

---

connUnitSnsPortIdentifier

Syntax	FcAddressId
Access	read-only
Status	mandatory
Description	The Port Identifier for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 3 }`

---

connUnitSnsPortName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Port WWN for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 4 }`

---

connUnitSnsNodeName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Node Name for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 5 }`

---

connUnitSnsClassOfSvc

Syntax	OCTET STRING (SIZE (1))

Access	read-only
Status	mandatory
Description	The Classes of Service offered by this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 6 }`

connUnitSnsNodeIPAddress

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The IPv6 formatted address of the Node for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 7 }`

connUnitSnsProcAssoc

Syntax	OCTET STRING (SIZE (1))
Access	read-only
Status	mandatory
Description	The Process Associator for this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 8 }`

connUnitSnsFC4Type

Syntax	OCTET STRING (SIZE (1))
Access	read-only
Status	mandatory
Description	The FC-4 Types supported by this entry in the SNS table.
Sequence	`::= { connUnitSnsEntry 9 }`

connUnitSnsPortType

Syntax	OCTET STRING (SIZE (1))
Access	read-only

Status mandatory
Description The Port Type of this entry in the SNS table.
Sequence `::= { connUnitSnsEntry 10 }`

---

connUnitSnsPortIPAddress

Syntax OCTET STRING (SIZE(16))
Access read-only
Status mandatory
Description The IPv6 formatted address of this entry in the SNS table.
Sequence `::= { connUnitSnsEntry 11 }`

---

connUnitSnsFabricPortName

Syntax FcNameId
Access read-only
Status mandatory
Description The Fabric Port name of this entry in the SNS table.
Sequence `::= { connUnitSnsEntry 12 }`

---

connUnitSnsHardAddress

Syntax FcAddressId
Access read-only
Status mandatory
Description The Hard ALPA of this entry in the SNS table.
Sequence `::= { connUnitSnsEntry 13 }`

---

connUnitSnsSymbolicPortName

Syntax DisplayString (SIZE (0..79))
Access read-only
Status mandatory
Description The Symbolic Port Name of this entry in the SNS table.

Sequence ::= { connUnitSnsEntry 14 }

connUnitSnsSymbolicNodeName

Syntax DisplayString (SIZE (0..79))

Access read-only

Status mandatory

Description The Symbolic Node Name of this entry in the SNS table.

Sequence ::= { connUnitSnsEntry 15 }

SNMP Trap Registration Group

trapMaxClients

Syntax INTEGER

Access read-only

Status mandatory

Description The maximum number of SNMP trap recipients supported by the connectivity unit.

Sequence ::= { trapReg 1 }

trapClientCount

Syntax INTEGER

Access read-only

Status mandatory

Description The current number of rows in the trap table.

Sequence ::= { trapReg 2 }

trapRegTable

Syntax SEQUENCE OF TrapRegEntry

Access not-accessible

Status mandatory

Description A table containing a row for each IP address/port number that traps will be sent to.

Sequence `::= { trapReg 3 }`

---

trapRegEntry

Syntax TrapRegEntry

Access not-accessible

Status mandatory

Description IP/Port pair for a specific client.

INDEX { trapRegIpAddress,
trapRegPort }

Sequence `::= { trapRegTable 1 }`

TrapRegEntry ::=
SEQUENCE {
trapRegIpAddress
IpAddress,
trapRegPort
INTEGER (1..2147483647),
trapRegFilter
FcEventSeverity,
trapRegRowState
INTEGER }

---

trapRegIpAddress

Syntax IpAddress

Access read-only

Status mandatory

Description The IP address of a client registered for traps.

Sequence `::= { trapRegEntry 1 }`

trapRegPort

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	The UDP port to send traps to for this host. Normally this would be the standard trap port (162). This object is an index and must be specified to create a row in this table.
Sequence	`::= { trapRegEntry 2 }`

trapRegFilter

Syntax	FcEventSeverity
Access	read-write
Status	mandatory
Description	This value defines the trap severity filter for this trap host. The connUnit will send traps to this host that have a severity level less than or equal to this value. The default value of this object is 'warning'.
Sequence	`::= { trapRegEntry 3}`

trapRegRowState

Syntax

INTEGER {

rowDestroy(1),	Remove row from table.
rowInactive(2),	Row exists, but TRAPs disabled
rowActive(3)	Row exists and is enabled for sending traps

}

Access: read-write

Status: mandatory

Description: Specifies the state of the row.

rowDestroy
READ: Can never happen.

WRITE: Remove this row from the table.

rowInactive
READ: Indicates that this row does exist, but that traps are not enabled to be sent to the target.

WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are not enabled to be sent to the target. If the row already existed, then traps are disabled from being sent to the target.

rowActive
READ: Indicates that this row exists, and that traps are enabled to be sent to the target.

WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are enabled to be sent to the target. If the row already exists, then traps are enabled to be sent to the target.

A value of rowActive or rowInactive must be specified to create a row in the table.

Sequence `::= { trapRegEntry 4}`

Related Traps

connUnitStatusChange

ENTERPRISE	fcmgmt
VARIABLES	{ connUnitStatus, connUnitState }
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
TYPE	FA MIB: Connection unit status change.
SUMMARY	FA MIB: Connection unit status change.
SEVERITY	Minor
CATEGORY	Status Events
TIMEINDEX	1
STATE	OPERATIONAL
Sequence	`::= 1`

NOTE: connUnitAddedTrap , 2, no longer used

connUnitDeletedTrap

ENTERPRISE	fcmgmt
VARIABLES	{ connUnitId }
Description	A connUnit has been deleted from this agent. Recommended severity level (for filtering): warning.
TYPE	FA MIB: Connection unit deleted.
SUMMARY	FA MIB: Connection unit deleted.
SEVERITY	Warning
CATEGORY	Status Events
TIMEINDEX	1
STATE	OPERATIONAL
Sequence	::= 3

connUnitEventTrap

ENTERPRISE	fcmgmt
VARIABLES	{ connUnitEventId, connUnitEventType, connUnitEventObject, connUnitEventDescr }
Description	An event has been generated by the connectivity unit. Recommended severity level (for filtering): info.
TYPE	FA MIB: Connection unit event trap.
SUMMARY	FA MIB: Connection unit event trap.
SEVERITY	Normal
CATEGORY	Configuration Events
TIMEINDEX	1
STATE	OPERATIONAL

Sequence	::= 4

---

connUnitSensorStatusChange

ENTERPRISE	fcmgmt
VARIABLES	{ connUnitSensorStatus }
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
TYPE	FA MIB: Connection unit sensor status change.
SUMMARY	FA MIB: Connection unit sensor status change.
SEVERITY	Minor
CATEGORY	Status Events
TIMEINDEX	1
STATE	OPERATIONAL
Sequence	::= 5

---

connUnitPortStatusChange

ENTERPRISE	fcmgmt
VARIABLES	{ connUnitPortStatus, connUnitPortState }
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
TYPE	FA MIB: Port status change.
SUMMARY	FA MIB: Port status change.
SEVERITY	Minor
CATEGORY	Status Events
TIMEINDEX	99
STATE	DEGRADED
Sequence	::= 6

END

G

McDATA/Nishan FC Fabric Element MIB

McDATA/Nishan FC Fabric Element Management MIB

Fabric Element Management MIB, Version 1.10 (same as version 1.9), as per Internet Draft, "Definitions of Managed Objects for the Fabric Element in Fibre Channel Standard", <draft-teow-fabric-element-mib-03.txt>, July 10, 1998.

This is edited for Nishan Systems based on public implementations.

Revision history:

4/20/01	Minor corrections to compile without warnings in SMICng.
5/08/03	Convert to v2 SMI for SMICng checking.
5/03/04	Define fibreChannel as a branch here.

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

FCFABRIC-ELEMENT-MIB DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE, experimental, Counter32, Gauge32, TimeTicks

FROM SNMPv2-SMI;

fibreChannel OBJECT IDENTIFIER ::= { experimental 42 }

fcFabric MODULE-IDENTITY

Last-Updated 0509190000Z

Organization McDATA Corporation

Contact-Info McDATA Corporation

380 Interlocken Crescent

Broomfield, CO 80021 USA

Tel : 1 720 558-8000

Fax : 1 720 558-3860

email : information@mcdata.com

Description The MIB for managing Fibre Channel Elements.

Revision 0509190000Z

Description Released for E/OSi version 4.7.

Sequence `::= { experimental 42 2 }`

fcFabric OBJECT IDENTIFIER ::= { fibreChannel 2 }

Fabric Element

fcFeMIBObjects OBJECT IDENTIFIER ::= { fcFabric 1 }

Groups under fcFe

fcFeConfig	OBJECT IDENTIFIER	::= { fcFeMIBObjects 1 }
fcFeStatus	OBJECT IDENTIFIER	::= { fcFeMIBObjects 2 }
fcFeError	OBJECT IDENTIFIER	::= { fcFeMIBObjects 3 }
fcFeAccounting	OBJECT IDENTIFIER	::= { fcFeMIBObjects 4 }
fcFeCapabilities	OBJECT IDENTIFIER	::= { fcFeMIBObjects 5 }

Type definitions.

DisplayString	::= OCTET STRING (SIZE (0..255))
MilliSeconds	::= INTEGER (0..2147383647) -- 2^31 - 1
MicroSeconds	::= INTEGER (0..2147383647)
FcNameId	::= OCTET STRING (SIZE (8))

Worldwide Name or Fibre Channel Name associated with an FC entity. It's a Network_Destination_ID or Network_Source_ID composed of a value up to 60 bits wide, occupying the remaining 8 bytes while the first nibble identifies the format of the Name_Identifier with hex values:

0: ignored

1: IEEE 48-bit address,

2: IEEE extended,

3: Locally assigned,

4: 32-bit IP address

FabricName ::= FcNameId

The Name Identifier of a Fabric. Each Fabric shall provide a unique Fabric Name. Only the following formats are allowed: IEEE48, and Local.

FcPortName ::= FcNameId

The Name Identifier associated with a port. Only the following formats are allowed: IEEE48, IEEE extended, and Local.

FcAddressId ::= OCTET STRING (SIZE (3)

Fibre Channel Address Identifier. A 24-bit value unique within the address space of a Fabric

FcRxDataFieldSize ::= INTEGER (128..2112)

Receive Data_Field Size

FcBbCredit ::= INTEGER (0..32767)

Buffer-to-buffer Credit

FC-PH version

FcphVersion ::= INTEGER (0..255)

Class 1 Stacked Connect Support/Mode
FcStackedConnMode ::= INTEGER { none (1), transparent (2), lockedDown (3) }

Class of Service Capability Set
FcCosCap ::= INTEGER (0..127)

bit 0	Class F
bit 1	Class 1
bit 2	Class 2
bit 3	Class 3
bit 4	Class 4
bit 5	Class 5
bit 6	Class 6
bit 7	reserved for future

FC-0 Baud Rates
Fc0BaudRate ::= INTEGER {

other (1),	none of below
oneEighth (2),	155 Mbaud (12.5MB/s)
quarter (4),	266 Mbaud (25.0MB/s)
half (8),	532 Mbaud (50.0MB/s)

full (16), 1 Gbaud (100MB/s)
double (32), 2 Gbaud (200MB/s)
quadruple (64) 4 Gbaud (400MB/s)) }

Baud Rate Capability Set
Fc0BaudRateCap ::= INTEGER (0..127)

bit 0	other
bit 1	oneEighth
bit 2	quarter
bit 3	half
bit 4	full
bit 5	double
bit 6	quadruple
bit 7	reserved for future

FC-0 Media Capability Set
Fc0MediaCap ::= INTEGER (0..65535)

bit 0	unknown
bit 1	single mode fibre (sm)
bit 2	multi-mode fibre 50 micron (m5)
bit 3	multi-mode fibre 62.5 micron (m6)
bit 4	video cable (tv)
bit 5	miniature cable (mi)
bit 6	shielded twisted pair (stp)
bit 7	twisted wire (tw)
bit 8	long video (lv)
bits 9-15	reserved for future use

A specific FC-0 medium type associated with a port
Fc0Medium ::= INTEGER {

unknown (1), sm (2), m5 (4), m6 (8), tv (16), mi (32), stp (64), tw (128), lv (256) }

The FC-0 transmitter type of a port
Fc0TxType ::= INTEGER {

unknown (1),

longWaveLaser (2), (LL)

shortWaveLaser (3), (SL)

longWaveLED (4), (LE)

electrical (5), (EL)

shortWaveLaserNoOFC (6) (SN) }

The FC-0 distance range associated with a port transmitter
Fc0Distance ::= INTEGER { unknown (1), long (2), intermediate (3), short (4) }

Module and Port Capacity
FcFeModuleCapacity ::= INTEGER (1..256)

FcFeFxPortCapacity ::= INTEGER (1..256)

Module, FxPort and NxPort Index
FcFeModuleIndex ::= INTEGER (1..256)

FcFeFxPortIndex ::= INTEGER (1..256)

FcFeNxPortIndex ::= INTEGER (1..126)

Port Mode
FcFxPortMode ::= INTEGER { unknown (1), fPort (2), flPort (3) }

BB_Credit Model
FcBbCreditModel ::= INTEGER { regular (1), alternate (2) }

Configuration group

This group consists of scalar objects and tables. It contains the configuration and service parameters of the Fabric Element and the FxPorts. The group represents a set of parameters associated with the Fabric Element or an FxPort to support its NxPorts. Implementation of this group is mandatory.

fcFeFabricName

Syntax FabricName

Max-Access read-only instead of read-write

Status	current
Description	The Name_Identifier of the Fabric to which this Fabric Element belongs.
Sequence	`::= { fcFeConfig 1 }`

fcFeElementName

Syntax	FcNameId
Max-Access	read-only-- instead of read-write
Status	current
Description	The Name_Identifier of the Fabric Element.
Sequence	`::= { fcFeConfig 2 }`

fcFeModuleCapacity

Syntax	FcFeModuleCapacity
Max-Access	read-only
Status	current
Description	The maximum number of modules in the Fabric Element, regardless of their current state.
Sequence	`::= { fcFeConfig 3 }`

The Module Table

This table contains one entry for each module, information of the modules.

fcFeModuleTable

Syntax	SEQUENCE OF FcFeModuleEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each module in the Fabric Element, information of the modules.
Sequence	`::= { fcFeConfig 4 }`

fcFeModuleEntry

Syntax	FcFeModuleEntry
Max-Access	not-accessible
Status	current
Description	An entry containing the configuration parameters of a module. INDEX { fcFeModuleIndex }

Sequence	`::= { fcFeModuleTable 1 }`
	FcFeModuleEntry ::=
	SEQUENCE {
	fcFeModuleIndex
	FcFeModuleIndex,
	fcFeModuleDescr
	DisplayString,
	fcFeModuleObjectID
	OBJECT IDENTIFIER,
	fcFeModuleOperStatus
	INTEGER,
	fcFeModuleLastChange
	TimeTicks,
	fcFeModuleFxPortCapacity
	FcFeFxPortCapacity,
	fcFeModuleName
	FcNameId }

fcFeModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current

Description This object identifies the module within the Fabric Element for which this entry contains information. This value is never greater than fcFeModuleCapacity.

Sequence `::= { fcFeModuleEntry 1 }`

fcFeModuleDescr

Syntax DisplayString

Max-Access read-only

Status current

Description A textual description of the module. This value should include the full name and version identification of the module. It should contain printable ASCII characters.

Sequence `::= { fcFeModuleEntry 2 }`

fcFeModuleObjectID

Syntax OBJECT IDENTIFIER

Max-Access read-only

Status current

Description The vendor's authoritative identification of the module. This value may be allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides a straight-forward and unambiguous means for determining what kind of module is being managed.

For example, this object could take the value 1.3.6.1.4.1.99649.3.9 if vendor 'Neufe Inc.' was assigned the subtree 1.3.6.1.4.1.99649, and had assigned the identifier 1.3.6.1.4.1.99649.3.9 to its 'FeFiFo-16 PlugInCard.'

Sequence `::= { fcFeModuleEntry 3 }`

fcFeModuleOperStatus

Syntax INTEGER {

online (1), functional

	offline (2),	not available
	testing (3),	under testing
	faulty (4)	defective }
Max-Access	read-only	
Status	current	
Description	This object indicates the operational status of the module:	
	online(1)	the module is functioning properly;
	offline(2)	the module is not available;
	testing(3)	the module is under testing; and
	faulty(4)	the module is defective in some way.
Sequence	`::= { fcFeModuleEntry 4 }`	

fcFeModuleLastChange

Syntax	TimeTicks
Max-Access	read-only
Status	current
Description	This object contains the value of sysUpTime when the module entered its current operational status. A value of zero indicates that the operational status of the module has not changed since the agent last restarted.
Sequence	`::= { fcFeModuleEntry 5 }`

fcFeModuleFxPortCapacity

Syntax	FcFeFxPortCapacity
Max-Access	read-only
Status	current
Description	The number of FxPort that can be contained within the module. Within each module, the ports are uniquely numbered in the range from 1 to fcFeModuleFxPortCapacity inclusive. However, the numbers are not required to be contiguous.
Sequence	`::= { fcFeModuleEntry 6 }`

fcFeModuleName

Syntax	FcNameId
Max-Access	read-only instead of read-write
Status	current
Description	The Name_Identifier of the module.
Sequence	`::= { fcFeModuleEntry 7 }`

FxPort Configuration Table

This table contains, one entry for each FxPort, configuration parameters of the ports.

fcFxPortTable

Syntax	SEQUENCE OF FcFxPortEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each FxPort in the Fabric Element, configuration and service parameters of the FxPorts.
Sequence	`::= { fcFeConfig 5 }`

fcFxPortEntry

Syntax	FcFxPortEntry
Max-Access	not-accessible
Status	current
Description	An entry containing the configuration and service parameters of a FxPort. INDEX { fcFxPortModuleIndex, fcFxPortFxPortIndex }

Sequence	`::= { fcFxPortTable 1 }`
	FcFxPortEntry ::=
	SEQUENCE {
	fcFxPortModuleIndex

FcFeModuleIndex,

fcFxPortFxPortIndex

FcFeFxPortIndex,

fcFxPortName

FcPortName,

FxPort common service parameters

fcFxPortFcphVersionHigh

FcphVersion,

fcFxPortFcphVersionLow

FcphVersion,

fcFxPortBbCredit

FcBbCredit,

fcFxPortRxBufSize

FcRxDataFieldSize,

fcFxPortRatov

MilliSeconds,

fcFxPortEdtov

MilliSeconds,

FxPort class service parameters

fcFxPortCosSupported

FcCosCap,

fcFxPortIntermixSupported

INTEGER,

fcFxPortStackedConnMode

FcStackedConnMode,

fcFxPortClass2SeqDeliv

INTEGER,

fcFxPortClass3SeqDeliv

INTEGER,

Other configuration parameters
fcFxPortHoldTime

MicroSeconds,

fcFxPortBaudRate

Fc0BaudRate,

fcFxPortMedium

Fc0Medium,

fcFxPortTxType

Fc0TxType,

fcFxPortDistance

Fc0Distance }

---

fcFxPortModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxPortEntry 1 }`

---

fcFxPortFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	`::= { fcFxPortEntry 2 }`

fcFxPortName

Syntax	FcPortName
Max-Access	read-only
Status	current
Description	The name identifier of this FxPort. Each FxPort has a unique port name within the address space of the Fabric.
Sequence	`::= { fcFxPortEntry 3 }`

FxPort common service parameters

fcFxPortFcphVersionHigh

Syntax	FcphVersion
Max-Access	read-only
Status	current
Description	The highest or most recent version of FC-PH that the FxPort is configured to support.
Sequence	`::= { fcFxPortEntry 4 }`

fcFxPortFcphVersionLow

Syntax	FcphVersion
Max-Access	read-only
Status	current
Description	The lowest or earliest version of FC-PH that the FxPort is configured to support.
Sequence	`::= { fcFxPortEntry 5 }`

fcFxPortBbCredit

Syntax	FcBbCredit
Max-Access	read-only
Status	current

Description — The total number of receive buffers available for holding Class 1 connect-request, Class 2 or 3 frames from the attached NxPort. It is for buffer-to-buffer flow control in the direction from the attached NxPort (if applicable) to FxPort.

Sequence — `::= { fcFxPortEntry 6 }`

---

fcFxPortRxBufSize

Syntax — FcRxDataFieldSize

Max-Access — read-only

Status — current

Description — The largest Data_Field Size (in octets) for an FT_1 frame that can be received by the FxPort.

Sequence — `::= { fcFxPortEntry 7 }`

---

fcFxPortRatov

Syntax — MilliSeconds

Max-Access — read-only

Status — current

Description — The Resource_Allocation_Timeout Value configured for the FxPort. This is used as the timeout value for determining when to reuse an NxPort resource such as a Recovery_Qualifier. It represents E_D_TOV (see next object) plus twice the maximum time that a frame may be delayed within the Fabric and still be delivered.

Sequence — `::= { fcFxPortEntry 8 }`

---

fcFxPortEdtov

Syntax — MilliSeconds

Max-Access — read-only

Status — current

Description — The E_D_TOV value configured for the FxPort. The Error_Detect_Timeout Value is used as the timeout value for detecting an error condition.

Sequence — `::= { fcFxPortEntry 9 }`

FxPort class service parameters

fcFxPortCosSupported

Syntax	FcCosCap
Max-Access	read-only
Status	current
Description	A value indicating the set of Classes of Service supported by the FxPort.
Sequence	`::= { fcFxPortEntry 10 }`

fcFxPortIntermixSupported

Syntax	INTEGER {yes (1), no (2)}
Max-Access	read-only
Status	current
Description	A flag indicating whether or not the FxPort supports an Intermixed Dedicated Connection.
Sequence	`::= { fcFxPortEntry 11 }`

fcFxPortStackedConnMode

Syntax	FcStackedConnMode
Max-Access	read-only
Status	current
Description	A value indicating the mode of Stacked Connect supported by the FxPort.
Sequence	`::= { fcFxPortEntry 12 }`

fcFxPortClass2SeqDeliv

Syntax	INTEGER {yes (1),no (2)}
Max-Access	read-only
Status	current

Description A flag indicating whether or not Class 2 Sequential Delivery is supported by the FxPort.

Sequence `::= { fcFxPortEntry 13 }`

fcFxPortClass3SeqDeliv

Syntax INTEGER {yes (1),no (2)}

Max-Access read-only

Status current

Description A flag indicating whether or not Class 3 Sequential Delivery is supported by the FxPort.

Sequence `::= { fcFxPortEntry 14 }`

Other FxPort parameters

fcFxPortHoldTime

Syntax MicroSeconds

Max-Access read-only

Status current

Description The maximum time (in microseconds) that the FxPort shall hold a frame before discarding the frame if it is unable to deliver the frame. The value 0 means that the FxPort does not support this parameter.

Sequence `::= { fcFxPortEntry 15 }`

fcFxPortBaudRate

Syntax Fc0BaudRate

Max-Access read-only

Status deprecated

Description The FC-0 baud rate of the FxPort.

Sequence `::= { fcFxPortEntry 16 }`

fcFxPortMedium

Syntax	Fc0Medium
Max-Access	read-only
Status	deprecated
Description	The FC-0 medium of the FxPort.
Sequence	`::= { fcFxPortEntry 17 }`

fcFxPortTxType

Syntax	Fc0TxType
Max-Access	read-only
Status	deprecated
Description	The FC-0 transmitter type of the FxPort.
Sequence	`::= { fcFxPortEntry 18 }`

fcFxPortDistance

Syntax	Fc0Distance
Max-Access	read-only
Status	deprecated
Description	The FC-0 distance range of the FxPort transmitter.
Sequence	`::= { fcFxPortEntry 19 }`

Operation group

This group consists of tables that contains operational status and established service parameters for the Fabric Element and the attached NxPorts. Implementation of this group is mandatory.

FxPort Operation table

This table contains, one entry for each FxPort, the operational status and parameters of the FxPorts.

fcFxPortStatusTable

Syntax	SEQUENCE OF FcFxPortStatusEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each FxPort in the Fabric Element, operational status and parameters of the FxPorts.
Sequence	`::= { fcFeStatus 1 }`

fcFxPortStatusEntry

Syntax	FcFxPortStatusEntry
Max-Access	not-accessible
Status	current
Description	An entry containing operational status and parameters of a FxPort. INDEX { fcFxPortStatusModuleIndex, fcFxPortStatusFxPortIndex }

Sequence	`::= { fcFxPortStatusTable 1 }`
	FcFxPortStatusEntry ::=
	SEQUENCE {
	fcFxPortStatusModuleIndex
	FcFeModuleIndex,
	fcFxPortStatusFxPortIndex
	FcFeFxPortIndex,
	fcFxPortID
	FcAddressId,
	fcFPortAttachedPortName
	FcPortName,
	fcFPortConnectedPort
	FcAddressId,
	fcFxPortBbCreditAvailable
	Gauge32,

fcFxPortOperMode
FcFxPortMode,
fcFxPortAdminMode
FcFxPortMode }

---

fcFxPortStatusModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxPortStatusEntry 1 }`

---

fcFxPortStatusFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	`::= { fcFxPortStatusEntry 2 }`

---

fcFxPortID

Syntax	FcAddressId
Max-Access	read-only
Status	current
Description	The address identifier by which this FxPort is identified within the Fabric. The FxPort may assign its address identifier to its attached NxPort(s) during Fabric Login.
Sequence	`::= { fcFxPortStatusEntry 3 }`

fcFPortAttachedPortName

Syntax	FcPortName
Max-Access	read-only
Status	deprecated
Description	The port name of the attached N_Port, if applicable. If the value of this object is '0000000000000000'H, this FxPort has no NxPort attached to it. This variable has been deprecated and may be implemented for backward compability.
Sequence	`::= { fcFxPortStatusEntry 4 }`

fcFPortConnectedPort

Syntax	FcAddressId
Max-Access	read-only
Status	deprecated
Description	The address identifier of the destination FxPort with which this FxPort is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this FxPort is not engaged in a connection. This variable has been deprecated and may be implemented for backward compability.
Sequence	`::= { fcFxPortStatusEntry 5 }`

fcFxPortBbCreditAvailable

Syntax	Gauge32
Max-Access	read-only
Status	current
Description	The number of buffers currently available for receiving frames from the attached port in the buffer-to-buffer flow control. The value should be less than or equal to fcFxPortBbCredit.
Sequence	`::= { fcFxPortStatusEntry 6 }`

fcFxPortOperMode

Syntax	FcFxPortMode

Max-Access	read-only
Status	current
Description	The current operational mode of the FxPort.
Sequence	`::= { fcFxPortStatusEntry 7 }`

fcFxPortAdminMode

Syntax	FcFxPortMode
Max-Access	read-only-- instead of read-write
Status	current
Description	The desired operational mode of the FxPort.
Sequence	`::= { fcFxPortStatusEntry 8 }`

fcFxPortPhysTable

Syntax	SEQUENCE OF FcFxPortPhysEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each FxPort in the Fabric Element, physical level status and parameters of the FxPorts.
Sequence	`::= { fcFeStatus 3 }`

fcFxPortPhysEntry

Syntax	FcFxPortPhysEntry
Max-Access	not-accessible
Status	current
Description	An entry containing physical level status and parameters of a FxPort. INDEX { fcFxPortPhysModuleIndex, fcFxPortPhysFxPortIndex }

Sequence	`::= { fcFxPortPhysTable 1 }`
	FcFxPortPhysEntry ::=
	SEQUENCE {
	fcFxPortPhysModuleIndex

FcFeModuleIndex,
fcFxPortPhysFxPortIndex
FcFeFxPortIndex,
fcFxPortPhysAdminStatus
INTEGER,
fcFxPortPhysOperStatus
INTEGER,
fcFxPortPhysLastChange
TimeTicks,
fcFxPortPhysRttov
MilliSeconds }

fcFxPortPhysModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxPortPhysEntry 1 }`

fcFxPortPhysFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	`::= { fcFxPortPhysEntry 2 }`

fcFxPortPhysAdminStatus

Syntax	INTEGER {	
	online (1),	place port online
	offline (2),	take port offline
	testing (3)	initiate test procedures
	}	
Max-Access	read-write	
Status	current	
Description	The desired state of the FxPort. A management station may place the FxPort in a desired state by setting this object accordingly. The testing(3) state indicates that no operational frames can be passed. When a Fabric Element initializes, all FxPorts start with fcFxPortPhysAdminStatus in the offline(2) state. As the result of either explicit management action or per configuration information accessible by the Fabric Element, fcFxPortPhysAdminStatus is then changed to either the online(1) or testing(3) states, or remains in the offline state.	
Sequence	`::= { fcFxPortPhysEntry 3 }`	

fcFxPortPhysOperStatus

Syntax	INTEGER {	
	online (1),	Login may proceed
	offline (2),	Login cannot proceed
	testing (3),	port is under test
	linkFailure (4)	failure after online/testing
	Other values may be used to indicate diagnostic for failed test.	
	}	
Max-Access	read-only	
Status	current	
Description	The current operational status of the FxPort. The testing(3) indicates that no operational frames can be passed. If fcFxPortPhysAdminStatus is offline(2) then fcFxPortPhysOperStatus	

should be offline(2). If fcFxPortPhysAdminStatus is changed to online(1) then fcFxPortPhysOperStatus should change to online(1) if the FxPort is ready to accept Fabric Login request from the attached NxPort; it should proceed and remain in the link-failure(4) state if and only if there is a fault that prevents it from going to the online(1) state.

Sequence `::= { fcFxPortPhysEntry 4 }`

---

fcFxPortPhysLastChange

Syntax TimeTicks

Max-Access read-only

Status current

Description The value of sysUpTime at the time the FxPort entered its current operational status. A value of zero indicates that the FxPort's operational status has not changed since the agent last restarted.

Sequence `::= { fcFxPortPhysEntry 5 }`

---

fcFxPortPhysRttov

Syntax MilliSeconds

Max-Access read-only -- instead of read-write

Status current

Description The Receiver_Transmitter_Timeout value of the FxPort. This is used by the receiver logic to detect Loss of Synchronization.

Sequence `::= { fcFxPortPhysEntry 6 }`

The FxPort Fabric Login table

This table contains, one entry for each FxPort in the Fabric Element, the Service Parameters that have been established from the most recent Fabric Login, implicit or explicit.

---

fcFxloginTable

Syntax SEQUENCE OF FcFxloginEntry

Max-Access not-accessible

Status current

Description A table that contains, one entry for each FxPort in the Fabric Element, services parameters established from the most recent Fabric Login, explicit or implicit.

Sequence `::= { fcFeStatus 4 }`

---

fcFxloginEntry

Syntax FcFxloginEntry

Max-Access not-accessible

Status current

Description An entry containing service parameters established from a successful Fabric Login.

INDEX { fcFxloginModuleIndex, fcFxloginFxPortIndex, fcFxloginNxPortIndex }

Sequence `::= { fcFxloginTable 1 }`

FcFxloginEntry ::=

SEQUENCE {

fcFxloginModuleIndex

FcFeModuleIndex,

fcFxloginFxPortIndex

FcFeFxPortIndex,

fcFxloginNxPortIndex

FcFeNxPortIndex,

fcFxPortFcphVersionAgreed

FcphVersion,

fcFxPortNxPortBbCredit

FcBbCredit,

fcFxPortNxPortRxDataFieldSize

FcRxDataFieldSize,

fcFxPortCosSuppAgreed

FcCosCap,

fcFxPortIntermixSuppAgreed
INTEGER,
fcFxPortStackedConnModeAgreed
FcStackedConnMode,
fcFxPortClass2SeqDelivAgreed
INTEGER,
fcFxPortClass3SeqDelivAgreed
INTEGER,
fcFxPortNxPortName
FcPortName,
fcFxPortConnectedNxPort
FcAddressId,
fcFxPortBbCreditModel
FcBbCreditModel }

fcFxloginModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxloginEntry 1 }`

fcFxloginFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

Sequence `::= { fcFxloginEntry 2 }`

---

fcFxloginNxPortIndex

Syntax FcFeNxPortIndex

Max-Access read-only

Status current

Description The object identifies the associated NxPort in the attachment for which the entry contains information.

Sequence `::= { fcFxloginEntry 3 }`

---

fcFxPortFcphVersionAgreed

Syntax FcphVersion

Max-Access read-only

Status current

Description The version of FC-PH that the FxPort has agreed to support from the Fabric Login

Sequence `::= { fcFxloginEntry 4 }`

---

fcFxPortNxPortBbCredit

Syntax FcBbCredit

Max-Access read-only

Status current

Description The total number of buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames to be transmitted to the attached NxPort. It is for buffer- to-buffer flow control in the direction from FxPort to NxPort. The buffer-to-buffer flow control mechanism is indicated in the respective fcFxPortBbCreditModel. [1](23.6.2.2).

Sequence `::= { fcFxloginEntry 5 }`

---

fcFxPortNxPortRxDataFieldSize

Syntax FcRxDataFieldSize

Max-Access	read-only
Status	current
Description	The Receive Data Field Size of the attached NxPort. This is a binary value that specifies the largest Data Field Size for an FT_1 frame that can be received by the NxPort. The value is in number of bytes and ranges from 128 to 2112 inclusive.
Sequence	`::= { fcFxloginEntry 6 }`

fcFxPortCosSuppAgreed

Syntax	FcCosCap
Max-Access	read-only
Status	current
Description	A variable indicating that the attached NxPort has requested the FxPort for the support of classes of services and the FxPort has granted the request.
Sequence	`::= { fcFxloginEntry 7 }`

fcFxPortIntermixSuppAgreed

Syntax	INTEGER {yes (1),no (2)}
Max-Access	read-only
Status	current
Description	A variable indicating that the attached NxPort has requested the FxPort for the support of Intermix and the FxPort has granted the request. This flag is only valid if Class 1 service is supported.
Sequence	`::= { fcFxloginEntry 8 }`

fcFxPortStackedConnModeAgreed

Syntax	FcStackedConnMode
Max-Access	read-only
Status	current
Description	A variable indicating whether the FxPort has agreed to support stacked connect from the Fabric Login. This is only meaningful if Class 1 service has been agreed.

Sequence ::= { fcFxloginEntry 9 }

---

fcFxPortClass2SeqDelivAgreed

Syntax INTEGER {yes (1),no (2)}

Max-Access read-only

Status current

Description A variable indicating whether the FxPort has agreed to support Class 2 sequential delivery from the Fabric Login. This is only meaningful if Class 2 service has been agreed.

Sequence ::= { fcFxloginEntry 10 }

---

fcFxPortClass3SeqDelivAgreed

Syntax INTEGER {yes (1),no (2)}

Max-Access read-only

Status current

Description A flag indicating whether the FxPort has agreed to support Class 3 sequential delivery from the Fabric Login. This is only meaningful if Class 3 service has been agreed.

Sequence ::= { fcFxloginEntry 11 }

---

fcFxPortNxPortName

Syntax FcPortName

Max-Access read-only

Status current

Description The port name of the attached NxPort, if applicable. If the value of this object is '0000000000000000'H, this FxPort has no NxPort attached to it.

Sequence ::= { fcFxloginEntry 12 }

---

fcFxPortConnectedNxPort

Syntax FcAddressId

Max-Access read-only

Status current

Description The address identifier of the destination FxPort with which this FxPort is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this FxPort is not engaged in a connection.

Sequence `::= { fcFxloginEntry 13 }`

fcFxPortBbCreditModel

Syntax FcBbCreditModel

Max-Access read-only -- instead of read-write

Status current

Description This object identifies the BB_Credit model used by the FxPort. The regular model refers to the Buffer-to-Buffer flow control mechanism defined in FC-PH [1] is used between the F_Port and the N_Port. For FL_Ports, the Alternate Buffer-to-Buffer flow control mechanism as defined in FC-AL [4] is used between the FL_Port and any attached NL_Ports.

Sequence `::= { fcFxloginEntry 14 }`

Error group

This group consists of tables that contain information about the various types of errors detected. The management station may use the information in this group to determine the quality of the link between the FxPort and its attached NxPort. Implementation of this group is optional.

The FxPort Error table

This table contains, one entry for each FxPort in the Fabric Element, counters recording numbers of errors detected since the management agent re-initialized. The first 6 columnar objects after the port index corresponds to the counters in the Link Error Status Block ([1](29.8)).

fcFxPortErrorTable

Syntax SEQUENCE OF FcFxPortErrorEntry

Max-Access not-accessible

Status current

Description A table that contains, one entry for each FxPort, counters that record the numbers of errors detected.

Sequence `::= { fcFeError 1 }`

---

fcFxPortErrorEntry

Syntax FcFxPortErrorEntry

Max-Access not-accessible

Status current

Description An entry containing error counters of a FxPort.

INDEX { fcFxPortErrorModuleIndex, fcFxPortErrorFxPortIndex }

Sequence `::= { fcFxPortErrorTable 1 }`

FcFxPortErrorEntry ::=

SEQUENCE {

fcFxPortErrorModuleIndex

FcFeModuleIndex,

fcFxPortErrorFxPortIndex

FcFeFxPortIndex,

fcFxPortLinkFailures

Counter32,

fcFxPortSyncLosses

Counter32,

fcFxPortSigLosses

Counter32,

fcFxPortPrimSeqProtoErrors

Counter32,

fcFxPortInvalidTxWords

Counter32,

fcFxPortInvalidCrcs

Counter32,

fcFxPortDelimiterErrors
Counter32,
fcFxPortAddressIdErrors
Counter32,
fcFxPortLinkResetIns
Counter32,
fcFxPortLinkResetOuts
Counter32,
fcFxPortOlsIns
Counter32,
fcFxPortOlsOuts
Counter32 }

fcFxPortErrorModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxPortErrorEntry 1 }`

fcFxPortErrorFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	`::= { fcFxPortErrorEntry 2 }`

fcFxPortLinkFailures

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of link failures detected by this FxPort.
Sequence	`::= { fcFxPortErrorEntry 3 }`

fcFxPortSyncLosses

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of loss of synchronization detected by the FxPort.
Sequence	`::= { fcFxPortErrorEntry 4 }`

fcFxPortSigLosses

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of loss of signal detected by the FxPort.
Sequence	`::= { fcFxPortErrorEntry 5 }`

fcFxPortPrimSeqProtoErrors

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of primitive sequence protocol errors detected by the FxPort.
Sequence	`::= { fcFxPortErrorEntry 6 }`

fcFxPortInvalidTxWords

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of invalid transmission word detected by the FxPort.
Sequence	`::= { fcFxPortErrorEntry 7 }`

fcFxPortInvalidCrcs

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of invalid CRC detected by this FxPort.
Sequence	`::= { fcFxPortErrorEntry 8 }`

fcFxPortDelimiterErrors

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Delimiter Errors detected by this FxPort.
Sequence	`::= { fcFxPortErrorEntry 9 }`

fcFxPortAddressIdErrors

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of address identifier errors detected by this FxPort.
Sequence	`::= { fcFxPortErrorEntry 10 }`

fcFxPortLinkResetIns

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Link Reset Protocol received by this FxPort from the attached NxPort.
Sequence	`::= { fcFxPortErrorEntry 11 }`

fcFxPortLinkResetOuts

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Link Reset Protocol issued by this FxPort to the attached NxPort.
Sequence	`::= { fcFxPortErrorEntry 12 }`

fcFxPortOlsIns

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Offline Sequence received by this FxPort.
Sequence	`::= { fcFxPortErrorEntry 13 }`

fcFxPortOlsOuts

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Offline Sequence issued by this FxPort.
Sequence	`::= { fcFxPortErrorEntry 14 }`

Accounting Groups

(1) Class 1 Accounting Group,

(2) Class 2 Accounting Group, and

(3) Class 3 Accounting Group.

Each group consists of a table that contains accounting information for the FxPorts in the Fabric Element. Implementation of each group is optional.

The Class 1 Accounting table

This table contains, one entry for each FxPort in the Fabric Element, Counters for certain types of events occurred in the FxPorts since the management agent has re-initialized. Implementation of this group is optional.

Class1 is not supported by Nishan, table removed

The Class 2 Accounting table

This table contains, one entry for each FxPort in the Fabric Element, Counters for certain types of events occurred in the FxPorts since the the management agent has re-initialized. Implementation of this group is optional.

fcFxPortC2AccountingTable

Syntax	SEQUENCE OF FcFxPortC2AccountingEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each FxPort in the Fabric Element, Class 2 accounting information recorded since the management agent has re-initialized.
Sequence	`::= { fcFeAccounting 2 }`

fcFxPortC2AccountingEntry

Syntax	FcFxPortC2AccountingEntry

Max-Access not-accessible

Status current

Description An entry containing Class 2 accounting information for each FxPort.

INDEX { fcFxPortC2AccountingModuleIndex, fcFxPortC2AccountingFxPortIndex }

Sequence `::= { fcFxPortC2AccountingTable 1 }`

FcFxPortC2AccountingEntry ::=

SEQUENCE {

fcFxPortC2AccountingModuleIndex

FcFeModuleIndex,

fcFxPortC2AccountingFxPortIndex

FcFeFxPortIndex,

fcFxPortC2InFrames

Counter32,

fcFxPortC2OutFrames

Counter32,

fcFxPortC2InOctets

Counter32,

fcFxPortC2OutOctets

Counter32,

fcFxPortC2Discards

Counter32,

fcFxPortC2FbsyFrames

Counter32,

fcFxPortC2FrjtFrames

Counter32 }

---

fcFxPortC2AccountingModuleIndex

Syntax FcFeModuleIndex

Max-Access read-only

Status current

Description This object identifies the module containing the FxPort for which this entry contains information.

Sequence `::= { fcFxPortC2AccountingEntry 1 }`

---

fcFxPortC2AccountingFxPortIndex

Syntax FcFeFxPortIndex

Max-Access read-only

Status current

Description This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

Sequence `::= { fcFxPortC2AccountingEntry 2 }`

---

fcFxPortC2InFrames

Syntax Counter32

Max-Access read-only

Status current

Description The number of Class 2 frames received by this FxPort from its attached NxPort.

Sequence `::= { fcFxPortC2AccountingEntry 3 }`

---

fcFxPortC2OutFrames

Syntax Counter32

Max-Access read-only

Status current

Description The number of Class 2 frames delivered through this FxPort to its attached NxPort.

Sequence `::= { fcFxPortC2AccountingEntry 4 }`

fcFxPortC2InOctets

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Class 2 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.
Sequence	`::= { fcFxPortC2AccountingEntry 5 }`

fcFxPortC2OutOctets

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Class 2 frame octets, including the frame delimiters, delivered through this FxPort to its attached NxPort.
Sequence	`::= { fcFxPortC2AccountingEntry 6 }`

fcFxPortC2Discards

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Class 2 frames discarded by this FxPort.
Sequence	`::= { fcFxPortC2AccountingEntry 7 }`

fcFxPortC2FbsyFrames

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of F_BSY frames generated by this FxPort against Class 2 frames.
Sequence	`::= { fcFxPortC2AccountingEntry 8 }`

fcFxPortC2FrjtFrames

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of F_RJT frames generated by this FxPort against Class 2 frames.
Sequence	`::= { fcFxPortC2AccountingEntry 9 }`

The Class 3 Accounting Group

This table contains, one entry for each FxPort in the Fabric Element, Counters for certain types of events occurred in the FxPorts since the management agent has re-initialized. Implementation of this group is optional.

fcFxPortC3AccountingTable

Syntax	SEQUENCE OF FcFxPortC3AccountingEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each FxPort in the Fabric Element, Class 3 accounting information recorded since the management agent has re-initialized.
Sequence	`::= { fcFeAccounting 3 }`

fcFxPortC3AccountingEntry

Syntax	FcFxPortC3AccountingEntry
Max-Access	not-accessible
Status	current
Description	An entry containing Class 3 accounting information for each FxPort. INDEX { fcFxPortC3AccountingModuleIndex, fcFxPortC3AccountingFxPortIndex }
Sequence	`::= { fcFxPortC3AccountingTable 1 }`

```
FcFxPortC3AccountingEntry ::=
SEQUENCE {
fcFxPortC3AccountingModuleIndex
        FcFeModuleIndex,
fcFxPortC3AccountingFxPortIndex
        FcFeFxPortIndex,
fcFxPortC3InFrames
        Counter32,
fcFxPortC3OutFrames
        Counter32,
fcFxPortC3InOctets
        Counter32,
fcFxPortC3OutOctets
        Counter32,
fcFxPortC3Discards
        Counter32 }
```

---

fcFxPortC3AccountingModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxPortC3AccountingEntry 1 }`

---

fcFxPortC3AccountingFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current

Description — This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

Sequence — `::= { fcFxPortC3AccountingEntry 2 }`

---

fcFxPortC3InFrames

Syntax — Counter32

Max-Access — read-only

Status — current

Description — The number of Class 3 frames received by this FxPort from its attached NxPort.

Sequence — `::= { fcFxPortC3AccountingEntry 3 }`

---

fcFxPortC3OutFrames

Syntax — Counter32

Max-Access — read-only

Status — current

Description — The number of Class 3 frames delivered through this FxPort to its attached NxPort.

Sequence — `::= { fcFxPortC3AccountingEntry 4 }`

---

fcFxPortC3InOctets

Syntax — Counter32

Max-Access — read-only

Status — current

Description — The number of Class 3 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.

Sequence — `::= { fcFxPortC3AccountingEntry 5 }`

---

fcFxPortC3OutOctets

Syntax — Counter32

Max-Access	read-only
Status	current
Description	The number of Class 3 frame octets, including the frame delimiters, delivered through this FxPort to its attached NxPort.
Sequence	`::= { fcFxPortC3AccountingEntry 6 }`

fcFxPortC3Discards

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The number of Class 3 frames discarded by this FxPort.
Sequence	`::= { fcFxPortC3AccountingEntry 7 }`

Capability Group

The Capability Group consists of a table describing information about what each FxPort is inherently capable of operating or supporting.

A capability may be used, as expressed in its respective object value in the Configuration group. Implementation of this group is optional.

fcFxPortCapTable

Syntax	SEQUENCE OF FcFxPortCapEntry
Max-Access	not-accessible
Status	current
Description	A table that contains, one entry for each FxPort, the capabilities of the port within the Fabric Element.
Sequence	`::= { fcFeCapabilities 1 }`

fcFxPortCapEntry

Syntax	FcFxPortCapEntry
Max-Access	not-accessible

Status current

Description An entry containing the capabilities of a FxPort.

INDEX { fcFxPortCapModuleIndex, fcFxPortCapFxPortIndex }

Sequence `::= { fcFxPortCapTable 1 }`

FcFxPortCapEntry ::=

SEQUENCE {

fcFxPortCapModuleIndex

FcFeModuleIndex,

fcFxPortCapFxPortIndex

FcFeFxPortIndex,

fcFxPortCapFcphVersionHigh

FcphVersion,

fcFxPortCapFcphVersionLow

FcphVersion,

fcFxPortCapBbCreditMax

FcBbCredit,

fcFxPortCapBbCreditMin

FcBbCredit,

fcFxPortCapRxDataFieldSizeMax

FcRxDataFieldSize,

fcFxPortCapRxDataFieldSizeMin

FcRxDataFieldSize,

fcFxPortCapCos

FcCosCap,

fcFxPortCapIntermix

INTEGER,

fcFxPortCapStackedConnMode

FcStackedConnMode,

fcFxPortCapClass2SeqDeliv

INTEGER,

fcFxPortCapClass3SeqDeliv

INTEGER,

fcFxPortCapHoldTimeMax

MicroSeconds,

fcFxPortCapHoldTimeMin

MicroSeconds,

fcFxPortCapBaudRates

Fc0BaudRateCap,

fcFxPortCapMedia

Fc0MediaCap }

---

fcFxPortCapModuleIndex

Syntax	FcFeModuleIndex
Max-Access	read-only
Status	current
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	`::= { fcFxPortCapEntry 1 }`

---

fcFxPortCapFxPortIndex

Syntax	FcFeFxPortIndex
Max-Access	read-only
Status	current
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	`::= { fcFxPortCapEntry 2 }`

fcFxPortCapFcphVersionHigh

Syntax	FcphVersion
Max-Access	read-only
Status	current
Description	The highest or most recent version of FC-PH that the FxPort is capable of supporting.
Sequence	`::= { fcFxPortCapEntry 3 }`

fcFxPortCapFcphVersionLow

Syntax	FcphVersion
Max-Access	read-only
Status	current
Description	The lowest or earliest version of FC-PH that the FxPort is capable of supporting.
Sequence	`::= { fcFxPortCapEntry 4 }`

fcFxPortCapBbCreditMax

Syntax	FcBbCredit
Max-Access	read-only
Status	current
Description	The maximum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached NxPort.
Sequence	`::= { fcFxPortCapEntry 5 }`

fcFxPortCapBbCreditMin

Syntax	FcBbCredit
Max-Access	read-only
Status	current
Description	The minimum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached NxPort.

Sequence ::= { fcFxPortCapEntry 6 }

---

fcFxPortCapRxDataFieldSizeMax

Syntax FcRxDataFieldSize

Max-Access read-only

Status current

Description The maximum size in bytes of the Data Field in a frame that the FxPort is capable of receiving from its attached NxPort.

Sequence ::= { fcFxPortCapEntry 7 }

---

fcFxPortCapRxDataFieldSizeMin

Syntax FcRxDataFieldSize

Max-Access read-only

Status current

Description The minimum size in bytes of the Data Field in a frame that the FxPort is capable of receiving from its attached NxPort.

Sequence ::= { fcFxPortCapEntry 8 }

---

fcFxPortCapCos

Syntax FcCosCap

Max-Access read-only

Status current

Description A value indicating the set of Classes of Service that the FxPort is capable of supporting.

Sequence ::= { fcFxPortCapEntry 9 }

---

fcFxPortCapIntermix

Syntax INTEGER {yes (1),no (2)}

Max-Access read-only

Status current

Description A flag indicating whether or not the FxPort is capable of supporting the intermixing of Class 2 and Class 3 frames during a Class 1 connection. This flag is only valid if the port is capable of supporting Class 1 service.

Sequence `::= { fcFxPortCapEntry 10 }`

---

fcFxPortCapStackedConnMode

Syntax FcStackedConnMode

Max-Access read-only

Status current

Description A value indicating the mode of Stacked Connect request that the FxPort is capable of supporting.

Sequence `::= { fcFxPortCapEntry 11 }`

---

fcFxPortCapClass2SeqDeliv

Syntax INTEGER {yes (1),no (2)}

Max-Access read-only

Status current

Description A flag indicating whether or not the FxPort is capable of supporting Class 2 Sequential Delivery.

Sequence `::= { fcFxPortCapEntry 12 }`

---

fcFxPortCapClass3SeqDeliv

Syntax INTEGER {yes (1),no (2)}

Max-Access read-only

Status current

Description A flag indicating whether or not the FxPort is capable of supporting Class 3 Sequential Delivery.

Sequence `::= { fcFxPortCapEntry 13 }`

---

fcFxPortCapHoldTimeMax

Syntax MicroSeconds

Max-Access	read-only
Status	current
Description	The maximum holding time (in microseconds) that the FxPort is capable of supporting.
Sequence	`::= { fcFxPortCapEntry 14 }`

fcFxPortCapHoldTimeMin

Syntax	MicroSeconds
Max-Access	read-only
Status	current
Description	The minimum holding time (in microseconds) that the FxPort is capable of supporting.
Sequence	`::= { fcFxPortCapEntry 15 }`

fcFxPortCapBaudRates

Syntax	Fc0BaudRateCap
Max-Access	read-only
Status	deprecated
Description	A value indicating the set of baud rates that the FxPort is capable of supporting. This variable has been deprecated and may be implemented for backward compability.
Sequence	`::= { fcFxPortCapEntry 16 }`

fcFxPortCapMedia

Syntax	Fc0MediaCap
Max-Access	read-only
Status	deprecated
Description	A value indicating the set of media that the FxPort is capable of supporting. This variable has been deprecated and may be implemented for backward compability.
Sequence	`::= { fcFxPortCapEntry 17 }`

END

H

McDATA Management Traps MIB

McDATA Management Traps MIB

SNMP v1 Format, Version 1.0

Revision history:

4/20/01	Initial version created by extracting traps from NISHAN-MGT.mib.
4/24/01	Removed nSswLaAggPortsDownTrap and nSswLaAggPortsUpTrap, which were never implemented. See Springbok TIR #1619 and #1620.
6/18/01	Merged 1000, 2000, and 3000 Series traps into a single file.
7/13/01	Corrected enterprise ID and varbind list for all v1 traps,removed v2 traps (now in NISHAN-TRAPS-v2.mib).
9/26/01	Updated Saturn trap descriptions to match v2.0 implementation, to resolve TIR #1780 and others.
9/28/01	Redefined all Saturn traps to match latest v2.0 plans.
10/9/01	Restored 9/26 trap definitions, v2.0 won't include revised traps.
10/10/01	Revised Saturn trap definitions yet again, as more information about the real implementation becomes available. Removed trap 14 entirely.
10/30/01	Added the v2.1 Saturn trap changes that did not make it into v2.0 (same as 9/28/01 version).
3/27/02	Added the NVS trap changes.

6/19/02 Removed phase-1 NVS traps

10/10/02 Assigned NNM compatible severity levels so they could be categorized correctly.

03/12/03 Included support for IPS 5000 series.

03/04/03 Added support for E_Port traps

05/19/05 Removed old comments about Enterprise ID -- they no longer apply.

06/01/05 Improved description of configurationConflict trap.

09/19/05 Released for E/OSi 4.7.

NISHAN-TRAPS DEFINITIONS ::= BEGIN

NISHAN-TRAPS-v1.MIB

IMPORTS

OBJECT-Type

FROM RFC-1212

TRAP-Type

FROM RFC-1215

nishan

FROM NISHAN-SMI;

nishanTrapMsg is defined here for use in the trap variable binding list. nishanTrapMsg is not a true MIB variable and cannot be read by an SNMP GET.

nishanProducts	OBJECT IDENTIFIER	::= { nishan 3 }
nishanMgmt	OBJECT IDENTIFIER	::= { nishan 1 }
nishanMgtTraps	OBJECT IDENTIFIER	::= { nishanMgmt 12 }
DisplayString	::= OCTET STRING	

nishanTrapMsg

Syntax	DisplayString (SIZE (0..255))
Access	read-only
Status	mandatory

Description A textual log message included in many Nishan Systems enterprise-specific traps.

Sequence `::= { nishanMgtTraps 1 }`

Eclipse Series traps

The following traps are supported on the Eclipse SAN Routers. These are sent in SNMP v1 format. All traps contain a single DisplayString in the varbind list. The DisplayString is a log entry that describes the event.

nSswSNSTrap

Enterprise nishanProducts

Variables { nishanTrapMsg }

Description The trap sender has been elected as the primary SNS server.

Type New primary SNS server elected.

Summary %s

Arguments { 0 }

Severity NORMAL

Category Configuration Events

TimeIndex 1

State OPERATIONAL

Sequence `::= 1`

nSswFlashOperationTrap

Enterprise nishanProducts

Variables { nishanTrapMsg }

Description A new software image has been successfully installed in flash memory.

Type Software image installed.

Summary %s

Arguments { 0 }

Severity	NORMAL
Category	Configuration Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 3

---

nSswEnvVoltageUpperThreshTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	One of the power supply voltages has exceeded its maximum threshold.
Type	Maximum voltage threshold exceeded.
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 5

---

nSswEnvVoltageLowerThreshTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	One of the power supply voltages has dropped below its minimum threshold.
Type	Minimum voltage threshold passed.
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events

TimeIndex	1
State	OPERATIONAL
Sequence	::= 6

nSswEnvTempUpperThreshTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	The internal temperature has exceeded its maximum threshold.
Type	Maximum temperature threshold exceeded.
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 7

nSswEnvChasFanStatusTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	A fan's rotation rate dropped below the minimum threshold.
Type	Fan failed.
Summary	"%s"
Arguments	{ 0 }
Severity	WARNING
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 8

nSswEnvChasPowerSupplyStatusTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	A power supply has changed state. The DisplayString argument indicates whether the power supply status changed from up to down or down to up.
Type	Power supply status changed.
Summary	%s
Arguments	{ 0 }
Severity	WARNING
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 9`

nSswLaAggDownTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	A link Aggregator is no longer functioning. Either it has been manually disabled, or all member ports are down.
Type	Aggregated group is down.
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 10`

nSswLaAggUpTrap

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	A link Aggregator is functioning normally again. It may have been manually re-enabled, or one of the member ports is back up.
Type	Aggregated group is now up.
Summary	%s
Arguments	{ 0 }
Severity	NORMAL
Category	Configuration Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 11`

iFCP Gateway Traps

The following traps are supported only on the IPS Model 3300 and 4300 switches.

ifcpBackupDown

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	Sent by a primary iFCP port when it cannot be backed up by its configured backup port. The backup port may be unreachable or not responding, or the backup port may be unable to act as backup due to its configuration. This trap may be sent after the switch is reset, when the configuration is changed, or when an existing backup port becomes unreachable. This trap repeats when the backup connection is retried unsuccessfully. There is no interruption of storage traffic, but the primary iFCP port is no longer protected from failure. The text message in the variable binding list includes the IP address of the primary iFCP port that cannot be backed up, and the IP address of the port configured to be the backup. The cause of the failure (e.g., timeout, rejected, incorrect configuration) is not provided. There is no trap sent when the backup relationship is established successfully or re-established successfully.
Type	Backup Down
Summary	%s
Arguments	{ 0 }
Severity	WARNING
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 14`

ifcpBackupActivated

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }

Description	This trap is sent by a backup iFCP port when the backup port begins to activate its backup connections. This may be caused by the primary port becoming unreachable, or the primary port informing the backup port that the primary link has gone down. The backup port will attempt to establish all remote connections learned from the primary port. If there are any learned connections, this trap will be followed by other traps, defined below, to indicate the success or failure of each remote connection. The text message in the variable binding list of this trap includes the IP address of the primary port that has failed, and the IP address of the backup port that is reporting the primary port's failure.
Type	Backup Activated
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 15`

ifcpRemoteConnectionUp

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	This trap is sent by an iFCP port when it successfully establishes an iFCP connection to a remote SAN. If the port makes multiple remote connections, one trap is sent for each connection. The connection may be a normal connection or a backup connection. The text message in the variable binding list includes the IP address of the iFCP port making the connection, and the IP address of the remote end of the connection.
Type	Remote Connection Up
Summary	%s
Arguments	{ 0 }
Severity	NORMAL

Category	Configuration Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 16`

ifcpRemoteConnectionDown

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	This trap is sent by an iFCP port when an existing remote connection is lost or terminated normally, or when a connection attempt fails. If multiple connections are lost or terminated, or multiple concurrent connection attempts fail, one trap is sent for each connection. This trap repeats when connection retries fail. The text message in the variable binding list includes the IP address of the local iFCP port making the connection, and the IP address of the remote end of the connection. This trap does not specify the reason for the connection being down (e.g., timeout on existing connection, user configuration change, remote end rejects connection, etc.)
Type	Remote Connection Down
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 17`

rPortConfigurationChanged

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	This trap is sent when an R_Port configuration change is made by the user. Possible user initiated configuration changes are: 1) Zoning policies, 2) Interconnect modes, 3) Fabric IDs, 4) Domain IDs, 5) Port

Zoning option, or 6) Cleanup actions. For example, for 1) Zoning policies, the nishanTrapMsg may be: (R_Port 4) Config Change: Zone policy change.

Type	R_Port Zone policy changed.
Summary	%s
Arguments	{ 0 }
Severity	NORMAL
Category	Configuration Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 18`

rPortZoningChanged

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	This trap is sent when a zoning update is initiated on an FC switch. A zoning update is required when there is a mismatch between the SAN Router's zoning configuration, and that of a FC switch. This occurs if a zoning change is made from SAN Router , or a FC switch attempts to alter a zone that is under the control of a SAN Router.
Type	R_Port zoning changed on connected FC switch.
Summary	%s
Arguments	{ 0 }
Severity	WARNING
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 19`

rPortFailedtoEstablishISL

Enterprise	nishanProducts

Variables	{ nishanTrapMsg }
Description	This trap is sent when a r_port failed to establish an ISL link with another switch. The message within the trap will contain information about why the ISL link failed to establish. For example if it is due to RA_TOV mismatches, the message returned would be (R_Port 1) ELP error: ED_TOV or RA_TOV mismatch.
Type	R_Port failed to estable ISL.
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 20`

---

rPortFabricReconfiguration

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	This trap is sent when a fabric reconfiguration happens on an e_port. This could happen if a new FC switch is added to an existing fabric. Or a switch on the fabric sends out a an ELP. or requests a re-build of the fabric.
Type	E_Port fabric reconfiguration.
Summary	%s
Arguments	{ 0 }
Severity	WARNING
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 21`

rPortFabricCriticalError

Enterprise	nishanProducts
Variables	{ nishanTrapMsg }
Description	This trap is sent when a r_port detects a critical error on an FC Fabric. The message within the trap will contain informational string on the critical error and which port it was encountered on.
Type	R_Port Fabric Error.
Summary	%s
Arguments	{ 0 }
Severity	CRITICAL
Category	Error Events
TimeIndex	1
State	OPERATIONAL
Sequence	`::= 22`
	END

I

McDATA Products Object Identifier Tree

NISHAN-PRODUCTS.mib

McDATA Products Object Identifier Tree

Version 1.0

Revision history:

4/25/01 Added revision list to header comments.

5/03/01 Added ipS4000 series support

8/27/02 Added IPS 5000 series support

1/20/03 Added IPS 3350 support

3/07/03 Added IPS 3200 support

4/09/03 Modified to show new model numbers

9/29/03 Modified to show new model numbers

3/05/04 Modified for Eclipse 2600 series

03/29/05: Added IBM OEM version of 1620

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header

NISHAN-PRODUCTS DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY

FROM SNMPv2-SMI

nishan

FROM NISHAN-SMI;

nishanProducts MODULE-IDENTITY

Last-Updated	00509190000Z
Organization	McDATA Corporation
Contact-Info	McDATA Corporation
	380 Interlocken Crescent
	Broomfield, CO 80021 USA
	Tel : 1 720 558-8000
	Fax : 1 720 558-3860
	email : information@mcdata.com
Description	MIB for listing full set of products supported.
Revision	0509190000Z
Description	Released for E/OSi version 4.7.
Revision	0309290000Z
Description	Initial public release of this MIB module.
Sequence	`::= { nishan 3 }`

Object Identifier Assignments

Switch	Object	Type	Sequence
Nishan routers	soipRouter	OBJECT IDENTIFIER	::= { nishanProducts 1 }
	ips2000	OBJECT IDENTIFIER	::= { soipRouter 1 }
	ips2x00	OBJECT IDENTIFIER	::= { ips2000 1 }
Nishan Switches	soipSwitch	OBJECT IDENTIFIER	::= { nishanProducts 2 }
IPS 3000 series	ips3000	OBJECT IDENTIFIER	::= { soipSwitch 1 }
Nishan IPS3100	ips3100	OBJECT IDENTIFIER	::= { ips3000 1 }
Nishan IPS3300	ips3300	OBJECT IDENTIFIER	::= { ips3000 2 }
Nishan IPS3320 (formerly 3350)	ips3320	OBJECT IDENTIFIER	::= { ips3000 3 }
McDATA Eclipse 1620 (formerly IPS1620)	eclipse1620	OBJECT IDENTIFIER	::= { ips3000 4 }
IBM version of the Eclipse 1620	ibm1620	OBJECT IDENTIFIER	::= { ips3000 5 }
Nishan IPS 4000 series	ips4000	OBJECT IDENTIFIER	::= { soipSwitch 2 }
Nishan IPS4100	ips4100	OBJECT IDENTIFIER	::= { ips4000 1 }
Nishan IPS4300	ips4300	OBJECT IDENTIFIER	::= { ips4000 2 }
Eclipse 26xx series	eclipse26xx	OBJECT IDENTIFIER	::= { soipSwitch 3 }
Eclipse 2600	eclipse2600	OBJECT IDENTIFIER	::= { eclipse26xx 1 }
Eclipse 2640	eclipse2640	OBJECT IDENTIFIER	::= { eclipse26xx 2 }

Switch	Object	Type	Sequence
Nishan gateways	soipGateway	OBJECT IDENTIFIER	::= { nishanProducts 3 }
	ips1000	OBJECT IDENTIFIER	::= { soipGateway 1 }
	ips1x00	OBJECT IDENTIFIER	::= { ips1000 1 }

J

McDATA SAN Router SMI Definitions

NISHAN-SMI.MIB

McDATA SAN Router (formerly Nishan Systems) SMI definitions

Version 1.0.

Revision history:

4/25/01	Added revision list to header comments.
7/09/01	Added FcWWN and FcID textual conventions
7/11/01	Renamed FcWWN and FcID to WWNtype and FCIDtype to minimize changes to other MIBs.

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

```
NISHAN-SMI DEFINITIONS ::= BEGIN

IMPORTS

enterprises,

MODULE-IDENTITY
        FROM SNMPv2-SMI

TEXTUAL-CONVENTION
        FROM SNMPv2-TC ;

nishan MODULE-IDENTITY
```

Last-Updated	0507270000Z
Organization	McDATA Corporation
Contact-Info	McDATA Corporation
	380 Interlocken Crescent
	Broomfield, CO 80021 USA
	Tel : 1 720 558-8000
	Fax : 1 720 558-3860
	email : information@mcdata.com
Description	SMI (Structure of Management Information) for McDATA Eclipse SAN Routers.
Revision	0509190000Z
Description	Released for E/OSi version 4.7.
Revision	0507270000Z
Description	Replaced Nishan's CONTACT-INFO with McDATA's info.
Revision	0101010000Z
Description	Initial public release of this MIB module.
Sequence	`::= { enterprises 4369 }.`

WWNtype

	::= TEXTUAL-CONVENTION
Status	current
Description	Represents an 8-byte Fibre Channel World Wide Name (WWN). Equivalent to the FcNameId textual convention in RFC 2837.
Syntax	OCTET STRING (SIZE (8))

FCIDtype

	::= TEXTUAL-CONVENTION
Status	current
Description	Represents a 3-byte Fibre Channel ID. Also used for the3-byte SoIP Socket Number that is the IP Storage equivalent of the Fibre

ChannelID. FcID is equivalent to the FcAddressId textual convention in RFC 2837.

Syntax OCTET STRING (SIZE (3))

K

McDATA SNTP Configuration MIB

Nishan SNTP Configuration MIB

This MIB contains management objects for Nishan SNTP configuration. This MIB is supported on IPS Series 3000, 4000 and 5000 switches.

Version 1.0

Revision history:

05/06/03: initial version released.

05/08/03: Incorporate review comments.

After 05/05/05, changes are recorded by adding another REVISION clause to the MODULE-IDENTITY macro below. Changes are no longer recorded as comments in this header.

NISHAN-SNTP DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE

FROM SNMPv2-SMI

InetAddress, InetAddressType

FROM INET-ADDRESS-MIB

nishan

FROM NISHAN-SMI;

idbNishanSNTP MODULE-IDENTITY

Last-Updated 0509190000Z

Organization	McDATA Corporation
Contact-Info	McDATA Corporation
	380 Interlocken Crescent
	Broomfield, CO 80021 USA
	Tel: 1 720 558-8000
	Fax : 1 720 558-3860
	email : information@mcdata.com
Description	The private MIB for Nishan SNTP configuration.
Revision	0509190000Z
Description	Released for E/OSi version 4.7.
Revision	0305060000Z
Description	Initial public release of this MIB module.
Sequence	`::= { nishan 23 }`

SNTP Configuration

sntpAdminStatus

Syntax	INTEGER {enable(1), disable(2)}
Max-Access	read-write
Status	current
Description	{disable} This field is used to enable/disable the admin status of SNTP protocol.
Sequence	`::= { idbNishanSNTP 1 }`

sntpRole

Syntax	INTEGER { client(1), server(2) }
Max-Access	read-write
Status	current

Description { client }

Specifies whether the SNTP protocol is configured as a client SNTP or as a server SNTP. Client SNTP will synchronize it's clock to that of a Server. User must first disable SNTP (by setting the Admin status to disable) before changing SNTP roles.

Sequence `::= {idbNishanSNTP 2}`

---

sntpServerIpAddressType

Syntax InetAddressType

Max-Access read-write

Status current

Description The type of Inet address in sntpServerIpAddress.

Sequence `::= { idbNishanSNTP 3 }`

---

sntpServerIpAddress

Syntax InetAddress

Max-Access read-write

Status current

Description { 0 }

The IP address of the SNTP Server. When the switch is configured as a SNTP server, This value has no meaning.

Sequence `::= { idbNishanSNTP 4 }`

---

sntpTimeZone

Syntax INTEGER (-720..840)

Max-Access read-write

Status current

Description { 0 }

Specifies the timezone where the switch resides in. The timezone is represented in minutes with respect to GMT (0).

Sequence `::= { idbNishanSNTP 5 }`

sntpDaylightSavingsTime

Syntax	INTEGER {enable(1), disable(2)}
Max-Access	read-write
Status	current
Description	{ disable } Specifies whether the Daylight Savings time is observed or not. Value of disable(1) specifies that daylight savings time is not observed.
Sequence	`::= { idbNishanSNTP 6 }`

sntpClientState

Syntax	INTEGER { operational(1), error(2) }
Max-Access	read-only
Status	current
Description	{ operational } Specifies the operational state of the Client SNTP. Value of Error indicates that client encountered an operational error, such as not being able to contact Server.
Sequence	`::= { idbNishanSNTP 7 }`

END

Index

A

agents, introduction to 1-1

C

commands, SNMP 1-2
configurationChangeTime 2-93
connUnitConfigurationChangeTime 2-102
connUnitContact 2-104
connUnitControl 2-103
connUnitDomainId 2-100
connUnitEntry 2-94
connUnitEventCurrID 2-106
connUnitEventFilter 2-105
connUnitFabricID 2-106
connUnitGlobalId 2-96
connUnitId 2-96
connUnitInfo 2-103
connUnitLocation 2-105
connUnitMaxEvents 2-105
connUnitModuleId 2-103
connUnitName 2-103
connUnitNumEvents 2-105
connUnitNumLinks 2-106
connUnitNumports 2-98
connUnitNumRevs 2-102
connUnitNumSensors 2-101
connUnitNumZones 2-102
connUnitPrincipal 2-101
connUnitProduct 2-99
connUnitProxyMaster 2-100
connUnitRevsDescription 2-108
connUnitRevsEntry 2-107
connUnitRevsIndex 2-108
connUnitRevsRevId 2-108
connUnitRevsTable 2-107
connUnitRevsUnitId 2-108
connUnitSensorCharacteristic 2-112
connUnitSensorEntry 2-109
connUnitSensorIndex 2-110
connUnitSensorInfo 2-111
connUnitSensorMessage 2-111
connUnitSensorName 2-110
connUnitSensorStatus 2-111
connUnitSensorTable 2-109
connUnitSensorType 2-112
connUnitSensorUnitId 2-110
connUnitSn 2-99
connUnitState 2-98
connUnitStatus 2-99
connUnitStatusChangeTime 2-102
connUnitTable 2-93
connUnitTableChangeTime 2-93
connUnitType 2-98
connUnitUpTime 2-100
connUnitUrl 2-100
connUnitVendorId 2-106

F

fabric element management MIB
- class 1 accounting table 2-77
- class 2 accounting table 2-80
- class 3 accounting table 2-82
- description of 2-53
- FxPort capability table 2-84
- FxPort configuration table 2-60
- FxPort error table 2-75

FxPort fabric login table 2-70
FxPort operation table 2-66
FxPort physical level table 2-68
module table 2-58
objects defined in 2-58
predefined types 2-53
Fibre Alliance MIB 1-5
Fibre Channel Fabric Element (FCFE) MIB 1-5
Fx_Port Configuration Table 2-60

G

getnextrequest command 1-2
getrequest command 1-2
getresponse command 1-3

I

iSCSI Device Table D-3
iscsiDevAdminStatus D-8
iscsiDevAlias D-6
iscsiDeviceEntry D-3
iscsiDevicesCurrentEntries D-9
iscsiDevicesMaxEntries D-9
iscsiDeviceTable D-3
iscsiDevIndex D-4
iscsiDevIpAddress D-5
iscsiDevName D-4
iscsiDevNodeWWN D-7
iscsiDevPortWWN D-7
iscsiDevRole D-6
iscsiDevRowStatus D-8
iscsiDevSoIPSocket D-7
iscsiDevSwitchPort D-5
iscsiDevTcpPort D-5
iscsiInitiatorAutoRegister D-9

M

McDATA private enterprise MIBs 1-5
MIB II
IP Routing table 2-29
MIB objects
table description
atIfIndex 2-22
atNetAddress 2-23
atPhysAddress 2-22
FabricName 2-53
Fc0BaudRate 2-55
Fc0BaudRateCap 2-55
Fc0Distance 2-57
Fc0MediaCap 2-56
Fc0Medium 2-56
Fc0TxType 2-56
FcAddressId 2-54
FcBbCredit 2-54
FcBbCreditModel 2-58
FcCosCap 2-54
FcElementName 2-58
fcFabricName 2-58
FcFeFxPortCapacity 2-57
FcFeFxPortIndex 2-57
FcFeModuleCapacity 2-57, 2-58
fcFeModuleDescr 2-59
fcFeModuleFxPortCapacity 2-60
FcFeModuleIndex 2-57
FcFeModuleLastChange 2-60
fcFeModuleName 2-60
FcFeModuleObjectID 2-59
fcFeModuleOperStatus 2-59
FcFeNxPortIndex 2-57
fcFPortAttachedPortName 2-67
FcFPortConnectedPort 2-67
fcFxConfFxPortIndex 2-61
fcFxlogiFxPortIndex 2-70
FcFxlogiNxPortIndex 2-70
FcFxPortAddressIdErrors 2-76
FcFxPortAdminMode 2-68
FcFxPortBaudRate 2-65
FcFxPortBbCredit 2-62
FcFxPortBbCreditAvailable 2-67
fcFxPortBbCreditModel 2-74
fcFxPortC1AcctFxPortIndex 2-78
FcFxPortC1ConnTime 2-79
FcFxPortC1Discards 2-80
FcFxPortC1FbsyFrames 2-78
FcFxPortC1FrjtFrames 2-79
FcFxPortC1InConnections 2-78
FcFxPortC1InFrames 2-79
FcFxPortC1InOctets 2-80
FcFxPortC1OutConnections 2-78
FcFxPortC1OutFrames 2-79
FcFxPortC1OutOctets 2-80
fcFxPortC2AcctFxPortIndex 2-81
FcFxPortC2Discards 2-82

FcFxPortC2FbsyFrames 2-82
FcFxPortC2FrjtFrames 2-82
FcFxPortC2InFrames 2-81
FcFxPortC2InOctets 2-81
FcFxPortC2OutFrames 2-81
FcFxPortC2OutOctets 2-82
fcFxPortC3AcctFxPortIndex 2-83
FcFxPortC3Discards 2-84
FcFxPortC3InFrames 2-83
FcFxPortC3InOctets 2-83
FcFxPortC3OutFrames 2-83
FcFxPortC3OutOctets 2-84
FcFxPortCapBaudRates 2-88
FcFxPortCapBbCreditMax 2-85
FcFxPortCapBbCreditMin 2-85
FcFxPortCapClass2SeqDeliv 2-87
FcFxPortCapClass3SeqDeliv 2-87
FcFxPortCapCos 2-86
FcFxPortCapFcphVersionHigh 2-84
FcFxPortCapFcphVersionLow 2-85
fcFxPortCapFxPortIndex 2-84
FcFxPortCapHoldTimeMax 2-87
FcFxPortCapHoldTimeMin 2-88
fcFxPortCapIntermix 2-86
FcFxPortCapMedia 2-88
FcFxPortCapRxDataFieldSizeMax 2-85
FcFxPortCapRxDataFieldSizeMin 2-86
FcFxPortCapStackedConnMode 2-86
FcFxPortClass2SeqDeliv 2-64
FcFxPortClass2SeqDelivAgreed 2-73
FcFxPortClass3SeqDeliv 2-64
FcFxPortClass3SeqDelivAgreed 2-73
FcFxPortConnectedNxPort 2-74
FcFxPortCosSuppAgreed 2-71
FcFxPortCosSupported 2-63
FcFxPortDelimiterErrors 2-76
FcFxPortDistance 2-66
FcFxPortEdtov 2-63
fcFxPortErrorFxPortIndex 2-75
FcFxPortFcphVersionAgreed 2-71
FcFxPortFcphVersionHigh 2-61
FcFxPortFcphVersionLow 2-62
FcFxPortHoldTime 2-64
FcFxPortID 2-66
FcFxPortIntermixSuppAgreed 2-72
fcFxPortIntermixSupported 2-63
FcFxPortInvalidCrcs 2-76
FcFxPortInvalidTxWords 2-76
FcFxPortLinkFailures 2-75
FcFxPortLinkResetIns 2-77
FcFxPortLinkResetOuts 2-77
FcFxPortMedium 2-65
FcFxPortMode 2-58
FcFxPortName 2-61
FcFxPortNxPortBbCredit 2-71
FcFxPortNxPortName 2-73
FcFxPortNxPortRxDataFieldSize 2-71
FcFxPortOlsIns 2-77
FcFxPortOlsOuts 2-77
fcFxPortOperFxPortIndex 2-66
FcFxPortOperMode 2-68
FcFxPortPhysAdminStatus 2-69
fcFxPortPhysFxPortIndex 2-68
FcFxPortPhysLastChange 2-70
FcFxPortPhysOperStatus 2-69
FcFxPortPhysRttov 2-70
FcFxPortPrimSeqProtoErrors 2-76
FcFxPortRatov 2-62
FcFxPortRxBufSize 2-62
FcFxPortSigLosses 2-75
FcFxPortStackedConnMode 2-64
FcFxPortStackedConnModeAgreed 2-73
FcFxPortSyncLosses 2-75
FcFxPortTxType 2-65
FcNameId 2-53
FcphVersion 2-54
FcPortName 2-54
FcRxDataFieldSize 2-54
FcStackedConnMode 2-54
icmpInAddrMaskReps 2-37
icmpInAddrMasks 2-37
icmpInDestUnreachs 2-35
icmpInEchoReps 2-36
icmpInEchos 2-36
icmpInErrors 2-35
icmpInMsgs 2-35
icmpInParmProbs 2-36
icmpInRedirects 2-36
icmpInSrcQuenchs 2-36
icmpInTimeExcds 2-35
icmpInTimestampReps 2-37
icmpInTimestamps 2-36
icmpOutAddrMaskReps 2-39
icmpOutAddrMasks 2-39

icmpOutDestUnreachs 2-38
icmpOutEchoReps 2-39
icmpOutEchos 2-38
icmpOutErrors 2-37
icmpOutMsgs 2-37
icmpOutParmProbs 2-38
icmpOutRedirects 2-38
icmpOutSrcQuenchs 2-38
icmpOutTimeExcds 2-38
icmpOutTimestampReps 2-39
ifAdminStatus 2-18
ifDescr 2-16
ifIndex 2-16
ifInDiscards 2-19
ifInErrors 2-20
ifInNUcastPkts 2-19
ifInOctets 2-19
ifInUcastPkts 2-19
ifInUnknownProtos 2-20
ifLastChange 2-19
ifMtu 2-18
ifNumber 2-15
ifOperStatus 2-18
ifOutDiscards 2-21
ifOutErrors 2-21
ifOutNUcastPkts 2-20
ifOutOctets 2-20
ifOutQLen 2-21
ifOutUcastPkts 2-20
ifPhysAddress 2-18
ifSpecific 2-21
ifSpeed 2-18
ipAdEntAddr 2-28
ipAdEntBcastAddr 2-28
ipAdEntIfIndex 2-28
ipAdEntNetMask 2-28
ipAdEntReasmMaxSize 2-29
ipDefaultTTL 2-23
ipForwarding 2-23
ipForwDatagrams 2-24
ipFragCreates 2-27
ipFragFails 2-27
ipFragOKs 2-27
ipInAddrErrors 2-24
ipInDelivers 2-25
ipInDiscards 2-25
ipInHdrErrors 2-24
ipInReceives 2-23
ipInUnknownProtos 2-24
ipNetToMediaIfIndex 2-33
ipNetToMediaNetAddress 2-34
ipNetToMediaPhysAddress 2-34
ipNetToMediaType 2-34
ipOutDiscards 2-25
ipOutNoRoutes 2-26
ipOutRequests 2-25
ipReasmFails 2-26
ipReasmOKs 2-26
ipReasmReqds 2-26
ipReasmTimeout 2-26
ipRouteAge 2-32
ipRouteDest 2-29
ipRouteIfIndex 2-29
ipRouteInfo 2-33
ipRouteMask 2-32
ipRouteMetric1 2-29
ipRouteMetric2 2-30
ipRouteMetric3 2-30
ipRouteMetric4 2-30
ipRouteMetric5 2-33
ipRouteNextHop 2-30
ipRouteProto 2-31
ipRouteType 2-31
ipRoutingDiscards 2-35
snmpEnableAuthenTraps 2-52
snmpInASNParseErrs 2-47
snmpInBadCommunityNames 2-47
snmpInBadCommunityUses 2-47
snmpInBadValues 2-48
snmpInBadVersions 2-46
snmpInGenErrs 2-48
snmpInGetNexts 2-49
snmpInGetRequests 2-49
snmpInGetResponses 2-49
snmpInNoSuchNames 2-47
snmpInPkts 2-46
snmpInReadOnlys 2-48
snmpInSetRequests 2-49
snmpInTooBigs 2-47
snmpInTotalReqVars 2-48
snmpInTotalSetVars 2-49
snmpInTraps 2-50
snmpOutBadValues 2-50
snmpOutGenErrs 2-50

snmpOutGetNexts 2-51
snmpOutGetRequests 2-51
snmpOutGetResponses 2-51
snmpOutNoSuchNames 2-50
snmpOutPkts 2-46
snmpOutSetRequests 2-51
snmpOutTooBigs 2-50
snmpOutTraps 2-51
sysContact 2-13
sysDescr 2-12
sysLocation 2-14
sysName 2-14
sysObjectID 2-12
sysServices 2-14
sysUpTime 2-13
tcpActiveOpens 2-41
tcpAttemptFails 2-41
tcpConnLocalAddress 2-43
tcpConnLocalPort 2-44
tcpConnRemAddress 2-44
tcpConnRemPort 2-44
tcpConnState 2-42
tcpCurrEstab 2-42
tcpEstabResets 2-41
tcpInErrs 2-44
tcpInSegs 2-42
tcpMaxConn 2-40
tcpOutRsts 2-44
tcpOutSegs 2-42
tcpPassiveOpens 2-41
tcpRetransSegs 2-42
tcpRtoAlgorithm 2-39
tcpRtoMax 2-40
tcpRtoMin 2-40
udpInDatagrams 2-45
udpInErrors 2-45
udpLocalAddress 2-45
udpLocalPort 2-46
udpNoPorts 2-45
udpOutDatagrams 2-45

MIB-II
additional IP objects 2-34
additional TCP objects 2-44
address translation group 2-22
definition of 2-12
ICMP group 2-35
interfaces group 2-15
IP address tabler 2-28
IP address translation table 2-33
IP group 2-23
SNMP group 2-46
system group 2-12
TCP connection table 2-42
TCP group 2-39
UDP group 2-45
UDP listener table 2-45

MIBs
Fibre Alliance MIB 1-5
Fibre Channel Fabric Element (FCFE) MIB 1-5
introduction to 1-4
MIB-II 1-4
MIB-II, definition of 2-12
standard 1-4

MIBs support by McDATA 1-4

N

network management, introduction to 1-1

Nishan Extension MIB
Architecture Group Interface
nishanArchIfaceDuplexSpeedGet A-43
nishanArchIfaceDuplexSpeedSet A-42
nishanArchIfaceEnable A-41
nishanArchIfaceEntry A-38
nishanArchIfaceLink A-42
nishanArchIfaceLogicalPort A-39
nishanArchIfaceName A-40
nishanArchIfacePort A-39
nishanArchIfaceRXOctetsNoErr A-43
nishanArchIfaceRXPacketsNoErr A-44
nishanArchIfaceSTPEnable A-42
nishanArchIfaceTable A-38
nishanArchIfaceTXOctetsNoErr A-43
nishanArchIfaceTXPacketsNoErr A-43
nishanArchIfaceType A-40
nishanArchIfaceUnit A-39
Authenticating device logins A-18
nishanCommonAuthDevicesMaxRadiusServers A-19
nishanCommonAuthRadiusServerAddressType A-21
nishanCommonAuthRadiusServerCredential A-22

nishanCommonAuthRadiusServerEntry A-19
nishanCommonAuthRadiusServerIndex A-20
nishanCommonAuthRadiusServerIPAddress A-21
nishanCommonAuthRadiusServerRetries A-22
nishanCommonAuthRadiusServerTable A-19
nishanCommonAuthRadiusServerTimeout A-22
nishanCommonAuthRadiusServerType A-20
nishanCommonAuthRadiusServerUdpPort A-21
Authentication
nishanCommonAuthCommIndex A-14
nishanCommonAuthCommName A-14
nishanCommonAuthCommPerm A-15
nishanCommonAuthCommState A-15
nishanCommonAuthCommTable A-14
nishanCommonAuthCommTableEntriesMax A-13
nishanCommonAuthHostCommName A-17
nishanCommonAuthHostEntry A-16
nishanCommonAuthHostIndex A-16
nishanCommonAuthHostIpAddress A-17
nishanCommonAuthHostIpMask A-17
nishanCommonAuthHostName A-17
nishanCommonAuthHostState A-18
nishanCommonAuthHostTable A-16
nishanCommonAuthHostTableEntriesMax A-13
Authentication Group A-11
Chip Group A-44
nishanChipStub A-44
IP Connectivity A-9
nishanCommonIPGateAddress A-9
nishanCommonIPGateAddressOnNextReset A-11
nishanCommonIPIpAddress A-9
nishanCommonIPIpAddressOnNextReset A-11
nishanCommonIPMACAddr A-9
nishanCommonIPNetMask A-10
nishanCommonIPNetMaskOnNextReset A-11
nishanCommonIPStatus A-10
Load TFTP
nishanCommonLoadExecute A-33
nishanCommonLoadExecuteStatus A-33
nishanCommonLoadTftpAddress A-32
nishanCommonLoadTftpFileName A-32
nishanCommonLoadType A-32
Load TFTP Group A-32
Miscellaneous A-34
nishanCommonMiscBaud A-36
nishanCommonMiscBOOTPOnOff A-35
nishanCommonMiscDHCPOnOff A-35
nishanCommonMiscPassword A-36
nishanCommonMiscProductName A-36
nishanCommonMiscReset A-36
nishanCommonMiscSaveToNvm A-34
nishanCommonMiscSnmpSecurityOnOff A-34
nishanCommonMiscSpanOnOff A-35
Trap
nishanCommonTrapCommAuthentication A-27
nishanCommonTrapCommBridge A-27
nishanCommonTrapCommColdStart A-26
nishanCommonTrapCommEntry A-25
nishanCommonTrapCommIndex A-26
nishanCommonTrapCommLinkDown A-27
nishanCommonTrapCommLinkUp A-27
nishanCommonTrapCommName A-26
nishanCommonTrapCommOEMSpecific A-28
nishanCommonTrapCommRMON A-28
nishanCommonTrapCommState A-28
nishanCommonTrapCommTable A-25
nishanCommonTrapCommTableEntriesMax A-24
nishanCommonTrapDestCommName A-30

nishanCommonTrapDestEntry A-29
nishanCommonTrapDestIndex A-29
nishanCommonTrapDestIpAddress A-30
nishanCommonTrapDestIpMask A-31
nishanCommonTrapDestName A-30
nishanCommonTrapDestState A-31
nishanCommonTrapDestTable A-29
nishanCommonTrapDestTableEntries-Max A-24
nishanCommonTrapTest A-24
Trap Group A-23
Version Group
nishanCommonVerAppSWMinor A-8
nishanCommonVerBootSWMajor A-7
nishanCommonVerBootSWMinor A-8
nishanCommonVerMIBMajor A-7
nishanCommonVerMIBMinor A-7
Nishan iSCSI Configuration D-1
Nishan Management MIB
NishanOperStatus E-8
Port Configuration
Management Port configuration E-9
Port configuration E-9

P

private enterprise MIBs 1-5

R

revisionNumber 2-91
rPortConfigurationChanged H-10

S

setrequest command 1-3
SNMP
description of 1-2
introduction to 1-1
SNMP commands 1-2
standard MIBs 1-4
statusChangeTime 2-92
systemURL 2-92

T

traps
definition of 1-6
purpose of 1-6

U

uNumber 2-91

V

variables
how SNMP changes 1-3
introduction to 1-3